University of Southern Queensland

Faculty of Health, Engineering and Sciences

Monitoring Elderly Falls in the Home Using Computer Vision

A dissertation submitted by

Jabin Smith

in fulfilment of the requirements of

ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Honours) (Instrumentation, Control and Automation)

Submitted October 2021

Abstract

The population of the world is getting older on average and there are some suggestions that by 2066, the population of Australians over the age of 65 could be as high as 20%. Researchers have also found that more and more elderly people are choosing to spend their twilight years living in their own homes, a phenomenon dubbed 'aging in place'. One of the biggest hospitalisation injuries for the elderly is that which is caused by falling. Falls account for 40% of injury related death in the elderly and researchers have also found that the likelihood of death is vastly increased if the elderly person is unable to get back up after they fall. A term called the 'long lie' is used to explain this phenomenon and the aim of this research will be to create a fall detection technique that can detect the occurrence of the long lie and thus, prevent it.

This project has explored different types of research already conducted on capturing falls within the home and analysed what types of technology were used. Of these fall detection techniques, this project is focussed on using computer vision to detect falls. The method of image processing used for this project was foreground extraction and the fall detection algorithm utilised shape analysis of the foreground mask to determine if a fall occurred. This project explored the reliability and effectiveness of this type of algorithm and, after several iterations, an algorithm is presented that was able to detect the occurrence of a fall in most scenarios in the datasets provided. The final algorithm used a foreground detector provided by MATLAB's computer vision toolbox in conjunction with a blob detector that was able to analyse the foreground mask and produce outputs based on the mask. The fall detection algorithm then uses a combination of these outputs to determine if a fall has occurred. This algorithm is unique in that it uses a state-based system where if any of the fall conditions exist, the state will change to a fall state. The system will remain in this state until the algorithm has detected that the person has returned to the upright position. This provided a means to ensure that the algorithm could detect the occurrence of a long lie. There were, however, many occasions when the algorithm either incorrectly identified a fall or did not detect the fall at all. This was mainly due to problems with the foreground extraction which led to the conclusion that if a suitable foreground extraction is not produced, then fall detection is not going to be reliable.

This research also explores the influence of ethics and what impact computer vision systems have on society. It will find that the major concerns surrounding computer vision systems and other artificial intelligence (AI) is the threat to security and privacy. It was also discovered that very little has been done to produce ethical standards in the AI industry and that it is incumbent on all stakeholders to ensure that a moral set of guidelines are produced for the entire lifecycle of an AI product. The research will also show that trust in AI is achieved by ensuring that users understand, not only the benefits of the technology, but also the risks associated with it and how these risks are mitigated.

University of Southern Queensland

Faculty of Health, Engineering and Sciences

ENG4111 and 4112 Research Project

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering and Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitles "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and any other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Jabin Smith

Student Number [redacted]

Acknowledgements

Firstly, I would like to acknowledge my supervisor, Tobias Low, who has kept me on track with my algorithm development and provided the appropriate feedback when necessary. He was also able to offer valuable guidance for the development of the dissertation and other key deliverables.

Secondly, I want to acknowledge my employer, Chevron Australia, who have given me the time and resources to be able complete this thesis. In addition to this, they have provided me with financial support for the tuition fees throughout the last four years of my university studies. They have also assisted in finding placements for me with the company's engineering teams to gain the required level of experience I need to graduate.

Thirdly, I want to tribute this dissertation to my grandfather, Lewis William Smith, who has provided the source of inspiration. Lew passed away in 2012 at the age of 92 after suffering a fall in his bathroom when he was at home alone. He was discovered some hours later and taken to hospital where he would ultimately succumb to his injuries several days later.

The most acknowledgement goes to my wife, Rebecca, who has endured my part-time studies over the last nine years. Since I embarked on this journey at the start of 2013, she has suffered my anxiousness, mood-swings and temper tantrums but has also, at times, shared in my elation and joy. Her support throughout this journey has not gone unnoticed and I doubt I would have been able to complete my degree without her encouragement and support. Thank you and much love to you Bec!

Table of Contents

List of Figures

List of Tables

Glossary of Terms

Ageing in place	Phenomenon where elderly people are choosing to remain in their homes as they grow old.
C/C++	A common general purpose computer programming language.
Computer Vision	A field of artificial intelligence that allows computers to analyse and learn digital imagery.
Depth Image	Image where each pixel relates to a distance from the image plane to an object in the RGB image.
False negative	Incorrect detection of a fall scene.
False positive	Incorrect detection of a non-fall scene.
Foreground extraction	Process of segmenting an image to create the foreground mask.
Foreground mask	Binary image containing pixels that belong to moving objects in the scene.
GMM	Gaussian Mixture Model. A probabilistic model for representing normally distributed subpopulations.
HMM	Hidden Markov Model. A probabilistic model used for labelling images.
HSV	Hue Saturation Value provides a numerical readout of an image that corresponds to the colours contained in it.
Image binarization	Process of segmenting an image into the background and foreground.
Image Dataset	Collection of images that are used to train the algorithm.
Long lie	Extended period of time where an elderly person remains on the ground after a fall.
MATLAB	Programming software used for numeric computing and other computer science applications.
Occlusion	A partial or complete obstruction in an image.
RGB	Red, green and blue. The primary colours from which all other colours are derived.
Segmentation	Process of partitioning a digital image into multiple segments.
True negative	Correct detection of non-fall scene.
True positive	Correct detection of a fall scene.

Chapter 1 – Introduction

1.1 Background

It is safe to say that Australia's, and indeed the world's, population is rapidly ageing. In fact, data from the Australian Bureau of Statistics (ABS) shows that from 2001 until 2019 the percentage of the population over the age of 65 has increased from 12.5% to 15.9%, a rise of over 3%. Some states, however, showed significantly higher growth in this statistic, such as Tasmania which showed an increase of 6.3% (Australian Bureau of Statistics 2020). Further to this, prediction data by the ABS show that with high life expectancy, high fertility and high net overseas migration, the percentage of people aged over 65 could be as high as 20% by 2066 (Australian Bureau of Statistics 2020). Adding to this, many elderly people are now also choosing to remain in their own homes as they get older in a phenomenon known as 'ageing in place'(Anderson et al. 2018, Roy et al. 2018). While this phenomenon has a positive outcome in terms of relieving pressure on aged care institutions, it creates another issue in trying to service the health care needs of the elderly within their homes (El-Bendary et al. 2013). This is particularly risky when considering the likelihood of the elderly falling within their homes and carers not being present to help them get up or get them urgent medical attention if required.

Falling is one of the most damaging injuries an elderly person can suffer and is the leading cause of hospitalisation injury in people over the age of 65. Further to this, falls also account for 40% of injury related deaths in people over the age of 85 (Lord and Sherrington 2001). Attar et al. (2021) have also found in their research, that falls are a major contributor to death in elderly persons and is a primary factor for hospitalisation, especially in those that suffer from dementia. In their study of nearly 280 elderly persons, they found that the types of falls, for example, falling forward or backward, were evenly distributed and that no one fall type was significantly more prevalent than another. They did notice, however, that hip fractures were the leading type of fracture and that the most common location for a fall event was in the home. They also found that there was a significant association with fall occurrences to smoking and dementia.

As a result of often living alone, when the elderly fall, they are often incapable of getting themselves to their feet again. This leads to spending long periods in the place where they fell, further exacerbating their injuries. Lord and Sherrington (2001) define this extended period spent on the ground as the 'long lie' and found in their research that the incidence of the long lie results in a far greater chance of the casualty dying as a result of their injuries. A study by Wild et al. (1981) also found that the occurrence of the long lie increased the mortality of the casualty. They found that 11 of their 20 test subjects that suffered the long lie died within a short period after their fall event, while only 21 people passed away out of 105 that did not suffer from a long lie. The aim of the research and outcomes for the Engineering Research Project will be centred around the prevention of the long lie by detecting falls within the home and then raising the alarm. The detection of falls will primarily focus on using computer vision and the aim is to develop an effective algorithm to use within an embedded computer vision system.

Computer vision can be described as the technique that allows computers to understand the contents of digital images and videos. From this comprehension of image content, machines can then be used to automate tasks ordinarily undertaken by humans using their own vision system (Lauronen 2017). When it comes to fall detection using computer vision, there are several predominant methods. Gutiérrez et al. (2021) has conducted a comprehensive study of recent vision-based fall detection research and found that a multitude of different techniques were used. They discovered that some of the main methods used to analyse the image were foreground extraction, skeleton joint tracking using Microsoft Kinect, depth characterisation and person detection through Convolutional Neural Networks (CCN) with You Only Look Once (YOLO) technology. Methods for detection of falls were numerous and included shape analysis techniques such as bounding box ratio, ellipse orientation and ratio, linear and angular acceleration, motion history images and centroid velocity. There were also some researchers who used machine and deep learning systems for fall detection. This included techniques such as Support Vector Machines (SVM), k-nearest neighbour (knn), Hidden Markov Models (HMM) and Artificial Neural Networks (ANN). The image analysis method utilised in this research project will use foreground extraction and several shape analysis techniques will be investigated for the fall detection algorithm.

1.2 Objectives

Whilst the overall aim for this research project is to develop a working algorithm that can be used within an embedded system to detect falls and prevent the long lie, there are several other objectives that will need to be accomplished along the way. These objectives along with their rationale are listed below.

- Background research on methods of fall detection – this will need to be undertaken prior to any development of a fall detection algorithm. Previous works undertaken in this field will help understand how computer vision can detect human motion including falls.

- Develop initial fall detection algorithm – Using the previous research, the next objective will be to develop an algorithm that can be used to test fall image data. Whilst this pilot algorithm is not intended to be fully effective, it will give a preliminary starting point for testing and upon which a more robust algorithm can be developed. This objective also includes researching a suitable development environment and then learning and understanding how to use the environment to develop the algorithm.

- Algorithm testing – Using a known dataset, the initial algorithm will be tested, and its performance analysed. The results of this testing will be used as a baseline for future algorithm revisions.

- Algorithm refinement and retesting – The objective of this stage is to refine the algorithm until an acceptable level of reliability is found. At each stage of refinement, the algorithm will be retested against the same dataset. It is envisioned that there will be several iterations of the algorithm at this stage.

- Develop prototype (time permitting) – This objective will be a ‘stretch’ target provided there is enough time after an effective algorithm is developed and tested. It is anticipated that the algorithm will be able to be reproduced onto a microprocessor type development board with an attached camera. The prototype will then be tested in live scenarios to measure its performance.

Chapter 2 – Literature Review

When it comes to the detection of falls there have been many researchers that have attempted to find a definitive fall detection system which can automatically and accurately detect falls. Mubashir et al. (2013) has found that this research has centred around three main techniques for fall detection. Wearable sensors, where a user will wear a garment that has some type of sensor attached to it. Ambience sensors, where devices attempt to sense events through audio and visual data, and vision sensors, where computer vision is used detect a person and monitor their actions. Whilst the research in this project is focused on using computer vision for fall detection, a review into the research of all the fall detection techniques will be made below.

Computer vision is a wonderfully powerful tool that has multiple uses and can provide significant benefits to the community, however, when misused it can deliver a terribly negative impact. The second part of this literature review will focus on the ethical aspects of computer vision and how it relates to fall detection.

2.1 Fall detection techniques

2.1.1 Wearable Sensors

There are a myriad of research articles for fall detection of the elderly that focus on wearable technology with varying degrees of success. A wearable device that alerts emergency contacts via Bluetooth to a laptop was designed and created by Tomkun and Nguyen (2010). The system uses a tri-axial accelerometer and comes with visual, audible and vibration alert options to alert the wearer of an abnormal body tilt. The system performed with credible results, however, there were still some instances where the system mistook falls for non-fall events.

Li et al. (2009) propose a wearable system using both tri-axial accelerometers and gyroscopes which they claim will improve the reliability and accuracy of detection of falls. They hoped that this system would be able to differentiate between unintentional and intentional changes in posture. While they claim that their algorithm returns an accuracy of around 90%, there are some detection issues with the system unable to differentiate the wearer getting into bed and when a fall occurs against a wall with a sitting posture.

A wearable wrist-watch style fall detection device is proposed by Degen et al. (2005). The device carries two acceleration sensors which measure acceleration on all three axes and the in-built algorithm uses the resultant force, or norm, from the forces due to gravity and movement. From the acceleration norm, the velocity can be estimated and when the velocity reaches a certain threshold and an impact is detected and no movement is detected for a period, a fall is confirmed. Then the watch will sound an audible alarm. The results from this technology were mixed, when falling forward the watch was able to detect a fall 100% of the time, whereas falling backwards and to the side yielded a fall detection in approximately half of the tests.

Wang et al. (2014) created a wearable device that incorporates an accelerometer as well temperature, humidity, and a separate heart rate monitor. Real-time data from the sensors is processed by an onboard micro-controller unit (MCU) and a customer interface monitors the information from the MCU. To improve the reliability of the signal data being sent from the MCU to the receiving base station in another room, sensor access points are set up throughout the home. Like other wearable technology, falls are detected by accelerometer thresholds, however, in this case heart rate monitoring and lie angle are also used to detect if a fall has occurred. The authors of this research claim an accuracy of around 97.5% for this wearable device detection system.

Ramachandran et al. (2018) used a combination of risk category algorithms coupled with machine learning to improve the accuracy in fall detection. Risk category was determined by conducting surveys and finding 23 different risk factors. Based on the probability of each risk, low, medium and high-risk categories were created for each subject. They then conducted experiments with wearable sensor technology to gather data on falls first without the risk category information, then secondly with the risk category data. They found that with the risk category data applied to the fall algorithms, there were less cases of the fall detection system producing false falls.

Boutellaa et al. (2019) describe a system of multiple wearable sensors that uses a covariance matrix as a feature extractor on the fused raw signals and a nearest neighbour classifier system to determine a fall. Using the covariance matrix, a directional relationship is formed between the sensor vectors and then measured using a Euclidean metric and two geodesic metrics. The algorithm was then tested on two available datasets with accuracy around the 92% mark.

A fall detection system using machine learning algorithms is discussed by Vallabh et al. (2016). They implement accelerometers, gyroscopes and magnetometers, which are commonly found in modern smartphones, to collect body movement data and then classify the movement using a threshold decision tree. The paper goes on to test different machine learning algorithms other than the threshold decision tree to test their performance compared to the threshold decision tree. The machine learning methods investigated include Artificial Neural Network (ANN), Support Vector Machine (SVM), Least Squares Method (LSM), k-nearest neighbour(k-nn) and the Naive-Bayes Method. Each method was tested using a dataset obtained from a web based medical institution. The algorithms were required to differentiate between Activities of Daily Living (ADL) and falls. The research found that the k-nn method showed the greatest accuracy and was an improvement on the threshold decision tree.

Wearable devices do present a cost advantage and are not complicated to setup, however, they are deemed to be relatively intrusive (Mubashir et al. 2013). Whilst reasonable success in fall detection can be found with wearable devices, the main problem with the technology is getting the target audience to wear them. Research has found that old people regularly forget to wear the device so the technology becomes unreliable (El-Bendary et al. 2013).

2.1.2 Ambience Sensors

Liu et al. (2020) propose a system that detects motion using infrared array sensors and cite low cost, low complexity and high accuracy as benefits of this system. The system first detects the position of the person in the room by using bicubic interpolation and background subtraction on the infrared image to remove other heat sources. Then using time and detection area algorithms on the image data fall detection can be found. The system yielded fairly reliable results and future work should incorporate a larger detection area and greater sensor numbers should be employed.

Suryadevara et al. (2012a) developed a system to monitor the usage of home appliances and thus create a pattern of usage for elderly people in the home environment. The network incorporates sensors that communicate with each other and to a base station in a mesh network topology using Zigbee communication protocol. At the higher level, a software module collates the data from the low-level sensor and develops the behavioural pattern from which irregular behaviour can be detected. In further research, Suryadevara et al. (2012b) show how this sensor network data can be used to determine the 'Wellness' of the elderly person.

Daim and Lee (2020) use IR-UWB radar sensors to detect human motion. To detect human motion in the area of transmitted electromagnetic signals, IR-UWB sensors take advantage of diffraction, shadowing, scattering and reflection caused by a person moving. The received electromagnetic signals are analysed and any change in amplitude coupled with the time of arrival help detect the human motion. In order for the sensors to be used as motion detectors, however, the output needs to be manipulated. Daim and Lee (2020) have modelled this system by taking the received signal, digitising it through an analogue to digital convertor, then putting the digitised signal through a band pass filter. From this signal the amplitude and time delay are found and from the time delay, the distance to the object can be found. The results of their experiments found that IR-UWB sensors could accurately determine different objects at certain distances. They also found that at other distances the sensor system produced large errors.

A floor vibration detection system is applied in research by Alwan et al. (2006). The vibration is detected by a special vibration monitor which incorporates a piezo-electric sensor in contact with the floor. The principle of operation is that different movements around a room create unique vibration signatures. It is hypothesised then, that a falling person hitting the floor will have a significantly different vibration signature to normal activities and other objects hitting the floor. The device was tested experimentally using human like dummies and other common objects. The results of test confirmed that the device could successfully detect a human fall within a certain distance range.

Ambient sensors are found to be reasonably cost effective and considered less intrusive than the other two methods, however, they are subject to false alarms and generally have a low fall detection accuracy (Mubashir et al. 2013).

2.1.3 Vision Sensors

There has been a number of researchers that have been investigating the use of cameras and computer vision to detect falls in the elderly as well as human motion in general. Gutiérrez et al. (2021) has conducted a comprehensive review of fall detection systems using computer vision as they believe that no systematic review has taken place in this field. They found that most detection techniques utilise a three-step approach to their systems. The first being image pre-processing to optimise the image so that later stages have a clean image to look at. The next step is characterisation, where the features of the image are analysed and quantified so that they can be classified in the last stage of the process. As suggested, the third step is classification where the system classifies the movement within the image.

Töreyin et al. (2005) conducted research using Hidden Markov Models (HMM) to model human movement and detect falls in video footage. HMMs can be described as random probability models that are used to characterise systems with randomly changing variables (Gutiérrez et al. 2021). The research by Töreyin et al. (2005) also involved examining the addition of audio data to the video footage and seeing if that would improve the accuracy of the fall detection. The system works by creating a time series signal describing the motion of the person. This is built using a bounding box that has been determined by detection of motion with the video image. The time series signal is converted to a wavelet transform, which are better at ignoring the stationary parts of the signal, and then added to the HMM. Wavelet data of the audio signal is then fused with the HMM of the video to determine fall events. The results of the testing of this system show that while the video HMM can detect a fall, without the audio addition the results are susceptible to false positives. That is, falls are detected by the system when there was not actually a fall.

Huang et al. (2009) describe a system that utilises an omni-directional camera to monitor the health of the elderly in the home or health care institutions. Their research presents a system suitable for a less simplified environment where lighting intensity and static objects are accounted for. The results of testing show that the use of an omni-directional camera does increase the accuracy of camera-based fall detection systems when compared to systems using standard cameras.

A system that uses low cost digital cameras installed in the homes of elderly persons who are living alone is proposed by de Miguel et al. (2017). The system they developed utilises a Raspberry Pi microprocessor development board with a camera module designed for use with the Raspberry Pi. The detection algorithm uses a data model based on the angle, ratio and ratio derivative, where the ratio is ratio of the rectangular movement detection area, and the ratio derivative is measuring the change in this rectangular ratio over time. A background subtractor is used to define the contour of the subject and uses a Gaussian type of technique to 'learn' the environment and disregard static objects. A Kalman filter is also used on the image to remove noise and interference and can predict future image states based on past images. A Kalman filter is a method used to make estimations of variables more precise by combining several inaccurate variable observations. In this system it is used to track the person through the image sequence (Gutiérrez et al. 2021). Optical flow is also employed in this system and is used to detect motion from one frame to the next whilst removing static objects.

The designers have also included a system to alert relevant parties in the event of a fall using email and messaging services. The results of the testing showed that this system was around 96% accurate, however, improvements are required to allow for occlusions, state differentiation and changes in lighting intensity.

Similar to most other vision based fall detection systems, the system designed by Foroughi et al. (2008) employs background subtraction to detect moving regions in an image. However, this system analyses three main features of the image to detect the state of the subject in the image. The approximated ellipse is used to approximate the shape of the subject in the frame and the changing ratio of the ellipse determine what state the person is in. Projection histograms can represent the 2-D shape of the silhouette and temporal changes in head position can be tracked to differentiate between the subject's states. Testing this system proved that using the three vision detection methods provided reliable results including an accuracy of around 91%.

A camera node network fall detection system is proposed by Williams et al. (2010) and involves a series of cameras installed around the home all connected via ethernet back to a central processing node. As with previous research, the background subtraction method is used to create a foreground image of the moving subject. The research involved analysing three different methods of detection. The first is the bounding box ratio method where the image blob created by the background subtraction is bound by a rectangular box and if the aspect ratio is less than one, a fall is detected. The second detection method researched was Hidden Markov Models (HMM) which essentially involves training the detector through simulations to differentiate posture states. The last method the authors looked at was a more simplified HMM that trains the system by gathering 'normal' sequences. Any variations in new sequences will be detected by the system as something unusual and will then run some tests to determine if a fall has actually occurred.

The multi-modal features of a Microsoft Kinect sensor is used to detect human falls in research conducted by Tran et al. (2017). They use the skeletal modelling feature of the Kinect along with the RGB output to model human activity. Upon receival of every image frame, the skeleton-based detector is called and the vertical velocity and the height to the ground plane from the human centre is determined. Due to the variance in the 3D poses of the human body, skeletal information is not always available, so in this case the fall detection algorithm determines a motion map from the RGB image. A fall is detected in the skeleton mode when thresholds in the distance and velocity are met. When a skeleton image is not available, the motion map is improved with a kernel descriptor and input into a Support Vector Machine (SVM) that has been trained using previous datasets. An SVM is a set of supervised learning algorithms that are used for regression and classification (Gutiérrez et al. 2021).

A system that combines a wearable sensor with vision and also utilises the features of the Kinect sensor is the system proposed by Kwolek and Kepski (2016). A fall is detected by an accelerometer worn on the user which triggers a fuzzy inference system that is used to determine human movement and classify body pose. Essentially, the fuzzy system is used to verify that a fall has occurred. The system works by taking the Kinect depth image and subtracting it from a constantly updating background reference depth image. They use several parameters for the fuzzy logic to determine if a fall has occurred. The height to width ratio of the bounding

box that describes the person, the height of the box in the current frame to the max height of the box in previous frames, the distance of the bounding box centroid to the floor and the largest standard deviation from the centroid to x and y coordinates of the image. The fuzzy logic employs two Mandani engines and a Sugeno engine to differentiate between activities of daily living (ADL) and falls.

In earlier work by Kwolek and Kepski (2015) a similar system was developed as described above, however, instead of using fuzzy logic to determine the fall classification, a k-nearest neighbour (k-nn) classifier based on a collection of representative examples was used. The system was then tested using the UR fall detection dataset and results found for sensitivity, specificity, precision and accuracy.

Convolutional Neural Networks (CNN) is another machine learning technique researchers are employing to help detect falls in computer vision. Gutiérrez et al. (2021) explain CNNs as an Artificial Neural Network (ANN) that can extract local descriptors, or maps, out of a confined part of an image and use them to characterise the image contents. Both Núñez-Marcos et al. (2017) and Maldonado-Bascón et al. (2019) use CNN's in their fall detection algorithms. Núñez-Marcos et al. (2017) claim that CNN's can learn a set of features if enough training examples are given. They train a CNN with optical flow images. Optical flow algorithms represent the motion of objects between two image frames as displacement vector fields. They used a 3-step training process which starts with training on the Imagenet dataset which has 1000 classes and 14 million images. The CNN is then retrained with optical flow stacks on the UCF101 dataset consisting of 101 human actions on 13320 videos. The weights are then frozen in the CNN for the third stage which enables fine tuning of the remaining layers to yield the fall or no fall classification.

Maldonado-Bascón et al. (2019) use CNN's on images gained from a mobile assistive robot with an onboard camera. They use a You Only Look Once (YOLO) detector based on a CNN to detect a person in the frame then applied a Support Vector Machine (SVM) to classify the person as a fall or non-fall. The steps for detection is that a person is detected in the image frame by the YOLO detector and a bounding box is applied with the coordinates used as the feature extraction. Fall classification is then applied to the bounding box by the SVM based on the aspect ratio of the bounding box, the normalised bounding box width and the normalised bounding box lower edge. These normalised parameters help the SVM determine if a person has actually fallen, or if they are just resting.

Rougier et al. (2007) developed a novel method for fall detection by analysing a motion history image of the detected person in conjunction with the change in shape of the person. They argue that no serious fall occurs without significant change in movement and shape so the characteristics of the motion history image can be analysed for changes in movement. Like a lot of the methods discussed, they use foreground extraction to derive the human shape then an approximated ellipse is developed to yield information about the orientation and outline of the human form in the image. The motion history image is then quantified to represent the movement over recent frames in the image sequence. When the threshold for motion history is met, the algorithm analyses the standard deviation of the orientation angle and the ellipse axis ratio and if either of

these parameters meet a set threshold, a fall is detected. The motion history then determines if that person is motionless and in need of assistance.

Computer vision continues to be a field that fascinates many researchers due to its complexities and challenges. Whilst it has the potential to be used in many applications such as surveillance, medical, sports, behavioural biometrics, robotics and even art and entertainment, its main challenges are overcoming the effects of lighting and shadows, occlusions and variation in object data (Kale and Patil 2016).

2.2 Computer Vision Ethics

When developing any type of project that will be introduced into the public arena it is vital that all potential impacts, both good and bad, are considered for the entire lifecycle of the product. This is especially accurate for projects that are intending to use computer vision at the core of their developments. This leads to a requirement of an understanding of ethics and what ethical complications will affect computer vision systems. Ethics can be described as the standards of behaviour expected from society and is based on social conventions for morality, that is, what is perceived as right or wrong by the communities we live in (Lauronen 2017). Coupland et al. (2009) and Lauronen (2017) cite Quinn (2004) who described ethics as a rational examination into people's moral beliefs and behaviours and when applied to technology, there is a need to comprehensively understand the impact to these standards.

The benefits of artificial intelligence (AI) are many and in an article from Standards Australia (2019) in 2016 there were 314,000 applications for inventions with over 1.6 million research papers. Further to this, AI spending is predicted to be nearly USD$80 billion in 2022 and contribute over USD$15 billion by 2030. They predict that AI will enhance the economic and social well-being and Coupland et al. (2009) believe that AI has the ability to enhance the quality of life for the elderly using it for supporting ambient assisted living (AAL). With all technology, however, there is always the potential for it to be used for more sinister and malicious purposes. Senior et al. (2003) argued back at the start of this century that computer vision applications were becoming more ubiquitous, particularly in the public space with surveillance camera technology. Even at that time, they were able to present issues where automated algorithms were being used to obtain information from surveillance camera installations. They believed that this information could be used to acquire details about the identity of a person and other privacy intrusive information. PricewaterhouseCoopers (2020) commissioned a guidance paper for business leaders in the computer vision industry and recognised that facial recognition technology can be discriminatory and intrusive. They went on to highlight a method used by the Chinese government that uses facial recognition along with gait recognition technology in surveillance video to identify a person. Skirpan and Yeh (2017) also highlight some concerning cases with the use of computer vision systems. They state an instance where a computer vision application showed discriminative and biased behaviour when undertaking facial recognition of dark-skinned people and firmly believe this has led to unwarranted and prejudicial attention from law enforcement agencies.

There are many concerning ethical categories when referring to AI and computer vision applications. Lauronen (2017) has categorised the ethical issues facing computer vision into six different categories. Espionage, where data is collected by spying without consent. Identity theft, where images are classified as containing personal data. Malicious attacks, where systems are broken into to spread malicious content. Copyright infringement is the use of intellectual property without the right to. Discrimination is the incorrect identification of a person's colour, gender or race, for example, and misinformation is regulating media communication, e.g. photos, to not represent something truthfully. In their research, Blank (2019) argues that due to the rapid expansion of computer vision and machine learning, new technology is being developed without the consideration of ethics. They present three main issues with computer vision and machine learning. Human rights, namely how we use a person's identity. Error rates, how we deal with false positives while ensuring accuracy of the system and bias, where we deal with incorrect classification of people. Standards Australia (2019) also identified bias as one of the key issues with AI systems as well trust in the systems, market dominance, privacy and security while Coupland et al. (2009) identifies the management of the data collection, the informed consent of the person the data being collected on, the privacy of the person, the surveillance methods and user involvement in development as being key issues for computer vision systems. PricewaterhouseCoopers (2020) add that corporations need to develop their own ethics and regulation and list interpretability of the system by the user as well as robustness and security of the system as significant concerns. They also declare that internal governance needs to be implemented to ensure ethical concerns are being dealt with appropriately. Skirpan and Yeh (2017) have also identified similar issues to the other researchers, however they add spoofing which is getting automated systems to act nefariously to inputs, and the effect of psychological harms where people become anxious from the constant feeling of being watched. The American Civil Liberties Union (ACLU) is also concerned about the effect of computer vision, particularly when used in surveillance applications. They worry also that automated algorithms can be used for sinister intentions and have listed criminal abuse, institutional abuse, abuse for personal purposes, discriminatory targeting and voyeurism where surveillance applications can be compromised (Senior et al. 2003).

The consequences of a computer vision application that does not consider its ethical impacts can be considerable and if not addressed, human rights violations could arise (Blank 2019). Coupland et al. (2009) argue that the elderly are particularly more vulnerable to the malicious use of computer vison due to their physicality and lack of technical know-how. Skirpan and Yeh (2017) speculate that there a multitude of possibilities where computer vision applications adversely affect the end user of the technology. Events such as online infrastructure being brought down by hacked IoT cameras, online photos being used by external agencies for unintended purposes and invasion of privacy from captured personal videos are all possibilities that impact the integrity of computer vision. Senior et al. (2003) have concerns about the data that is available from computer vision systems. They worry about what data is present in these systems and who can see the data without consent from the owner plus the length of time data is stored for. The stored videos may also contain other 'metadata' which further puts the privacy of the user at risk.

Coupland et al. (2009) highlight the moral issues coming from the development of computer vision and how they will impact the lives of the elderly. The believe that all computer vison applications should be developed for the good of the user and issues concerning ethics during development and after development need to be identified. Due to the rapid speed in AI development, Blank (2019) has found that little regard has been given to the consequences of the technology and that time for reflection is required to take stock of what developments exist and what impacts they can have on society. They also state that without addressing these ethical issues, trust will be lost, meaning the technology may suffer a public backlash. They also point out that an 'ethical infrastructure' needs to be developed amongst all the stakeholders of AI. This responsibility is incumbent on governments, technology developers and society in general. Government agencies need to build the regulations and standards and provide incentives for following them. Technology developers must produce their own internal standards and society must set the expectations for AI technology and ensure that transparency is maintained (Blank 2019). Skirpan and Yeh (2017) also believe that new computer vision technology needs to be developed with a moral compass and that researchers and engineers need to be responsible for establishing the norms, otherwise legislation and market forces will not be quick enough to avoid the risk. As of 2019, standardisation of AI is in its infancy globally and Standards Australia (2019) believe that AI will benefit from the introduction of standards. They believe that these standards will help the community benefit from the digital economy and efficiency and productivity will be expanded. They expect that interoperability can then be considered and that standards will ensure the overall quality of the product. Despite no presence of a specific computer vision standard, in the UK, the Data Protection Privacy Act 1998 contains the principles for which data should be secured. They mention principles such as not keeping the data any longer than necessary, processing the data for specific and limited purposes, fairly and legally processing the data and that the data is processed in accordance with subjects' rights. Internationally, the Institution for Electrical and Electronic Engineers (IEEE) have consulted with academic, government and industry institutions to release a series of papers in their Global Initiative on The Ethics of Autonomous and Intelligent Systems program. In this initiative they detail the key values to consider with AI developments. The OECD have also released a series of guiding principles that outline ethical development of AI and deal with issues such as growth, legal responsibilities, transparency, security and accountability (Standards Australia 2019).

There are several methods that can help improve the integrity of computer vision systems. Senior et al. (2003) present a model for video privacy that illustrates several privacy aspects of the technology, what data is present, consent, who sees the data, how long is the data kept, how raw is the data and what form is the data in. For each of these aspects they present methods that can be used to address them. To limit what data is present, a system can use unfocussed lens as well as low-resolution cameras. It is easier to gain consent in a private setting but much more difficult in the public environment. For this they suggest a system that uses data from the video to verify the person trying to access the video. They believe that data should be stored in the digital medium and have a high level of encryption applied to it as well as encryption during the capture stage. Access should also be limited to those who need to see it via access control methods with multiple access authorisation levels. Finally, they suggest that controlling how raw the data is can be the most effective way at preventing

privacy issues. By masking many of the details in the video, the amount of information that can be gathered from it is limited. Some more solutions for dealing with ethical threats are presented by Skirpan and Yeh (2017). They argue that licensing and professional certifications could be implemented for developers of systems where the impacts can have life changing consequences. The development of black box tests, auditing systems and unbiased datasets are also mentioned as methods to prevent possible ethical issues. They also suggest that third party groups test the systems to see if there are any weaknesses that can be exposed prior to implementing in the public space. They could also employ independent agencies that can act as certifiers of systems who would then apply their seal of approval once satisfied with the integrity.

The benefits of implementing a computer vision system that has considered all the ethical implications are obvious but Standards Australia (2019) argue that an AI system can benefit and enhance social wellbeing if it can be trusted at protecting privacy and maintaining security. The problem is, however, that an ethical system can be rather complex to develop. In their research, Blank (2019), found that implementation of an ethical AI system enables the developers to take advantage of the social impact the AI delivers and also means that they can avoid future costs from mistakes that weren't realised during the development phase. They highlight a framework method from AI4People that uses the field of bioethics applied to AI systems to help developers implement an ethical AI system. This framework incorporates how the technology will benefit society, ensuring the technology does no harm, that autonomy of the person is maintained and ensuring the system is fair and does not commit undue injustices. A fifth principle, outside of bioethics, is also applied that deals with accountability of the system and how well it can be explained to the user.

In finishing ethical considerations, the future of computer vision systems used in the public and private spaces will need to address all implications to society, whether they be positive or negative. This assessment needs to incorporate the entire lifecycle of the technology from concept, through development, into implementation and through to disposal.

Chapter 3 – Research Design and Methodology

3.1 Research Design

The literature review has shown that there have been several different methods for detecting falls using computer vision, however, all the methods of computer vision can be broadly gathered into several separate computer vision categories. One category is a method that uses image segmentation to separate foreground objects, which is usually the object of interest, from the background. This was traditionally done using background subtraction, where the image to be analysed is subtracted from a reference image. However, newer algorithms have recently been developed that use machine learning to create the foreground mask. de Miguel et al. (2017), Foroughi et al. (2008) and Rougier et al. (2007) all use the background subtraction method to define the region of interest in the image frame. Once image segmentation is complete, the features of the resulting blob can be analysed to determine if a fall has occurred. Image analysis and fall detection has also been done using more advanced techniques that fit into the domain of deep learning. Deep learning, when used in the context of computer vision, can generally be described as a system that uses artificial neural networks to analyse an image. By training the computer with a multitude of different images of a unique feature, for example images of fallen people, the neural networks begin to recognize those features and can start to learn the difference between an image with that feature and those without. Núñez-Marcos et al. (2017) and Maldonado-Bascón et al. (2019) both use deep learning techniques in their fall detection algorithms. Whilst deep learning is a field that warrants further research for fall detection, this project utilises image segmentation to separate the region of interest and then extract the features that might determine a fall event from that region. It is hoped that with further research after this project some of the deep learning techniques could be applied to improve the performance of the fall detection algorithm.

A common method for determining a fall from a segmented image has been using the features of the bounding box that surrounds the person in an image generated by the image analysis software. Töreyin et al. (2005) use the bounding box to create a time series that describes the motion of the person in the image and de Miguel et al. (2017) use the ratio and ratio derivative of the bounding box sides to define the movement of the person over time. Williams et al. (2010) also use the aspect ratio of the bounding box so that if the ratio is less than one, a fall is detected. This method relies on the person falling in manner that is not perpendicular with the camera's horizontal axis. This method was also adopted by Kwolek and Kepski (2016) who used the ratio to verify a fall had occurred after detection by a wearable sensor. The original algorithm adopted in this project utilised the features of the bounding box generated by the region property function in MATLAB. To determine a fall, the ratio of the bounding box vertical lengths to the horizontal lengths were analysed and if the horizontal lengths were greater than the vertical, a fall was detected. It was found that this method had limitations when used as a fall detector by itself and would need additional features used in conjunction with it to become a reliable fall detection technique.

The characteristics of the human shape were used to verify a fall in the work by Rougier et al. (2007). To do this, an approximated ellipse was used to represent the human shape in a segmented image and the changes in angle and axis ratio helped determine if the person had fallen. This method has been utilised in conjunction with the bounding box ratio in the second iteration of the fall algorithm for this project. The MATLAB blob analyser function generates the information to create the ellipse and the centroid of the blob created by the foreground mask. The standard deviations of the ellipse angles and the axis ratios from the most recent frames are both used to help identify a fall in the image sequence.

This project hoped to construct some sort of prototype that could utilise the developed algorithm and be tested on real scenarios. The literature review showed that de Miguel et al. (2017) developed their fall detection system to be utilised on the Raspberry Pi platform with an attached camera. This project will hope to use a similar system in future development.

This project will attempt to build on some of the computer vision fall detection methods highlighted in the literature review and will hope to answer the following questions,

- Is foreground extraction suitable for use with fall detection algorithms?
- Is shape analysis for fall detection methods suitable?
- Can computer vision realistically be adopted as a fall detection method for elderly people?

3.2 Methodology

3.2.1 Algorithm Development

A qualitative approach was used for the consideration of the fall detection algorithm. Research has shown that, in the field of computer vision, there are many different methods for analysing the posture of humans and then using algorithms for ascertaining what that posture or change in posture means. This project has based its algorithm on and extended the work by Rougier et al. (2007) who uses human motion history plus other methods to establish if the image sequence has shown a fall. This research seemed easily understood and aspects of their methods have been used in this project.

Algorithm development was carried out using MATLAB software. The decision surrounding the choice of integrated development environment was made based on previous learned knowledge of MATLAB, as well as having access to appropriate licensing through USQ. Further to this, MATLAB can be easily converted into C/C++ computer language protocol for use in a microprocessor embedded system. MATLAB also has a suite of computer vision, image processing and deep learning tools that can be used to develop a reliable fall detection algorithm.

The initial stages of program development involved developing a method for image segmentation. By segmenting the image, a foreground mask that displays a person or any object as a white blob on a black

background is produced. Further processing was then conducted on the image to remove any unwanted features and provide a clean foreground mask. The features of the blob produced by the image segmentation are then analysed to produce several outputs that can be used to determine if the person in the image has indeed fallen.

The actual fall detection algorithm program has been separated from the main program to run as function when called by the main program. This has been done so that revisions made in the main program can easily be carried out without affecting the fall detection algorithm. For example, if the image segmentation was adjusted in the main program, the new parameters generated by this segmentation would be sent to the same fall detection algorithm and improvements or deficiencies could be quickly detected.

Another part of the program produces a video with added image annotations including shapes and text that show the viewer when the algorithm has detected a fall. Also, when a fall is detected in one or a series of images, the program adds a true value to the image label spreadsheet which was included with the fall image dataset. This value is then compared with the label for that particular image in the image label spreadsheet for evaluating the performance of the algorithm.

3.2.2 Algorithm Testing Requirements

The effectiveness of the algorithm must be understood to determine if the algorithm can not only reliably detect a fall, but also establish if the person has recovered or not. Further to this, the algorithm must show a high level of repeatability and be adaptable across different scenarios. For this, the system used by Kwolek and Kepski (2015) has been adopted where there are three measures used to determine how well the algorithm has performed – accuracy, sensitivity and specificity. The accuracy is a measure of how often the algorithm has correctly detected an actual fall (true positive) and an actual non-fall (true negative). The sensitivity is a measure of how often the algorithm has correctly detected an actual fall (true positive) compared to how often it missed a fall (false negative). Specificity, on the other hand, is a measure of how often the algorithm correctly detects a non-fall situation (true negative) compared to how often it incorrectly detected a fall (false positive).

For the algorithm to be deemed successful all performance measures must be satisfactory. No one measure can be high while the other two are low for the algorithm to be reasonably reliable. For example, if the sensitivity is high while the specificity and accuracy are low it would signal that the algorithm is detecting falls in a lot of frames including the frames where there are no falls present. For the purposes of this project, the performance measures would need to be upwards of 90% for the algorithm to be reliable. This threshold is based on the research conducted by Gutiérrez et al. (2021) for similar fall detection methods.

Further to this, the performance of the image segmentation must be analysed as well. This can only be done by visually checking the binary image output as the algorithm runs through the image sequence. This is necessary to ensure that falls that are detected are due to correct analysis of the human shape and not due to some random shapes produced by the image segmentation.

3.2.3 Testing the algorithm using datasets

As mentioned above, the algorithms that have been developed have been tested on publicly available datasets provided by Adhikari et al. (2017b). The authors have provided 20 datasets which contain over 22,000 images. Each dataset has a series of RGB and depth images which have been taken using a Microsoft Kinect camera. A label.csv file is also included which labels each of the images based on the different poses the subject person is performing in the image.

The bulk of the algorithm development was carried out using one dataset. When the algorithm was at the stage of testing it was tested on other datasets. However, not every dataset was suitable for the algorithm, for instance, if the subject person started in the first frame of the image sequence, a reliable foreground mask could not be established. The algorithm was able to be tested on the remaining appropriate datasets, however, and results were compared with the label spreadsheet.

Chapter 4 – Results and Discussion

4.1 Algorithm design

This chapter will detail the design of not only the fall detection algorithm, but also the image segmentation and pre-processing that was conducted to get the image to a stage that could be used in the algorithm. The programming of the image processing and algorithm went through several iterations before reaching the final outcome and at each stage, the algorithm was tested on the same datasets to determine if any changes had created an improvement in fall detection.

4.1.1 Initial image segmentation

Effective image segmentation is vital to the success of the fall detection algorithm, as a poorly segmented image will yield a shape that does not resemble the posture of the subject human in the frame. Many of the datasets acquired from Adhikari et al. (2017b) included a reference image, which was simply an image of the empty room. The initial image segmentation technique utilised a basic form of background subtraction where

Figure 4.1 Image segmentation using background subtraction

the subject image is subtracted from the reference image. The purpose of the background subtraction is so that any objects that are always present in the image will be ignored and only changes to the image, such as a person walking into frame, will be presented in the foreground mask. In lieu of a reference image, some of the datasets had no person in the initial image so this was used as the reference image. Unfortunately, some of the image datasets did not come with a reference image or had a person in the initial image, so these datasets were unable to be used for this algorithm.

After the initial background subtraction, the image is converted to grayscale and blurred to allow for better segmentation. The next step involves 'binarizing' the image where the image is processed so that any pixel above a certain threshold will be given a value of one, and the remaining pixels get a value of zero. This creates a raw binary mask where most of the white pixels, those with a value of one, should depict the outline of a shape that was not present in the reference image. In this case it should take on a humanoid shape. The image then has the MATLAB function bwareaopen, applied to it that 'opens' the image. This essentially removes any connected pixels below a size threshold and then any holes are filled in the remaining connected pixels so that the image is left with only the largest, connected blobs. A property filter function is then applied so that only the one largest blob is left in the image. Figure 4.1 shows an example of the blob after complete image segmentation.

A connected components function is then applied to the image that finds connected components within the image and returns a structure with fields that describe the component. A region property function is then applied to the connected components structure that can return multiple properties of the connected component such as, centroid location, area in pixels, bounding box and axis lengths. Properties from this function were used in the initial fall detection algorithm.

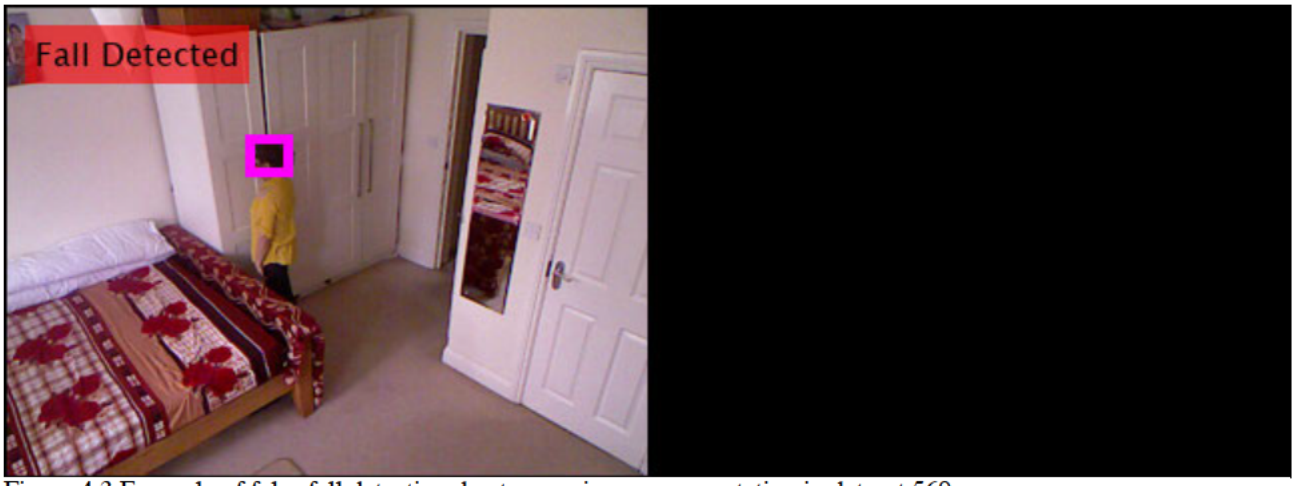

Figure 4.3 Example of false fall detection due to poor image segmentation in dataset 569

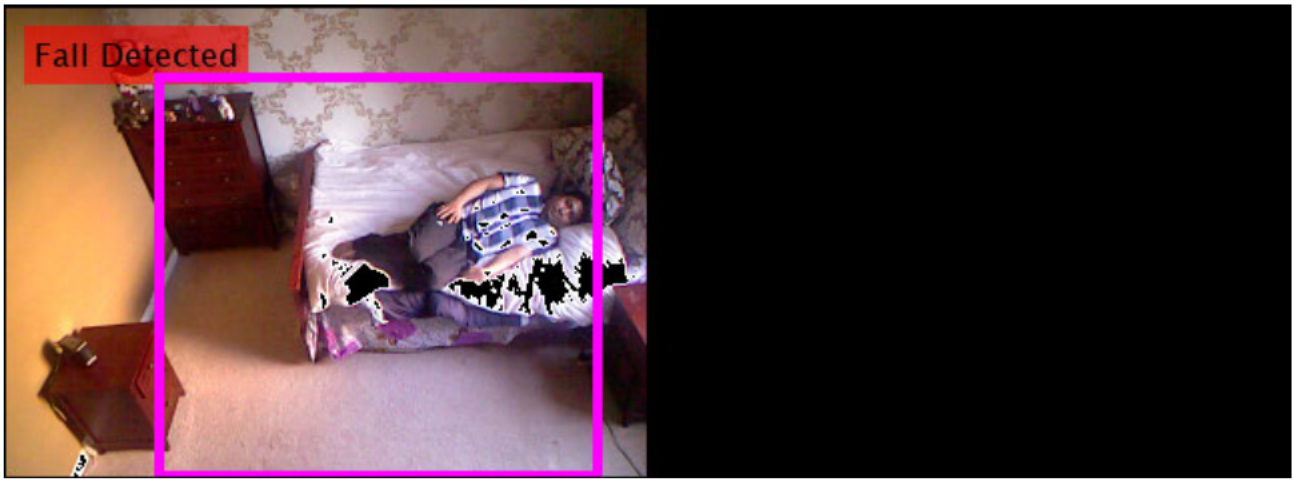

Figure 4.2 Poor image segmentation resulting in an inaccurate bounding box in dataset 581

4.1.2 Fall detection using bounding box ratio

One of the simplest methods of determining changes in the posture of a human is to analyse the changes in bounding box surrounding the human shape. For the initial fall detection algorithm, the vertical length was compared to the horizontal length and if the ratio was less than one, a fall was detected. The process is described in the flow chart in Figure 4.4 and it was shown that while this detection method was often able to detect a change of state to a lying pose, there were a number of times it was unable to detect a fall due to the bounding box ratio staying above one. Further to this, image segmentation using the background subtraction method was not very effective, particularly in poor lighting conditions and where the contrast between the foreground and background was low. This meant that the bounding box ratio would sometimes change to a value below one when the subject person was in a standing position. Figure 4.3 shows an example of a poorly segmented image from dataset 569 that has created a false fall detection using background subtraction segmentation method. At other times image segmentation would produce spurious blobs that would make the bounding box appear larger than the subject person. Figure 4.2 from image dataset 581 shows an example of when the image segmentation has produced a blob that doesn't accurately represent the human subjects shape. It is worth noting that despite the inaccurate bounding box size the ratio is lower than one and the algorithm has correctly detected a fall, however, this type of fault has the potential to skew the performance of the algorithm and give results that may seem legitimate when, in reality, it is not so. This is why it is important when measuring the performance of the algorithm, that the quality of the image segmentation is considered.

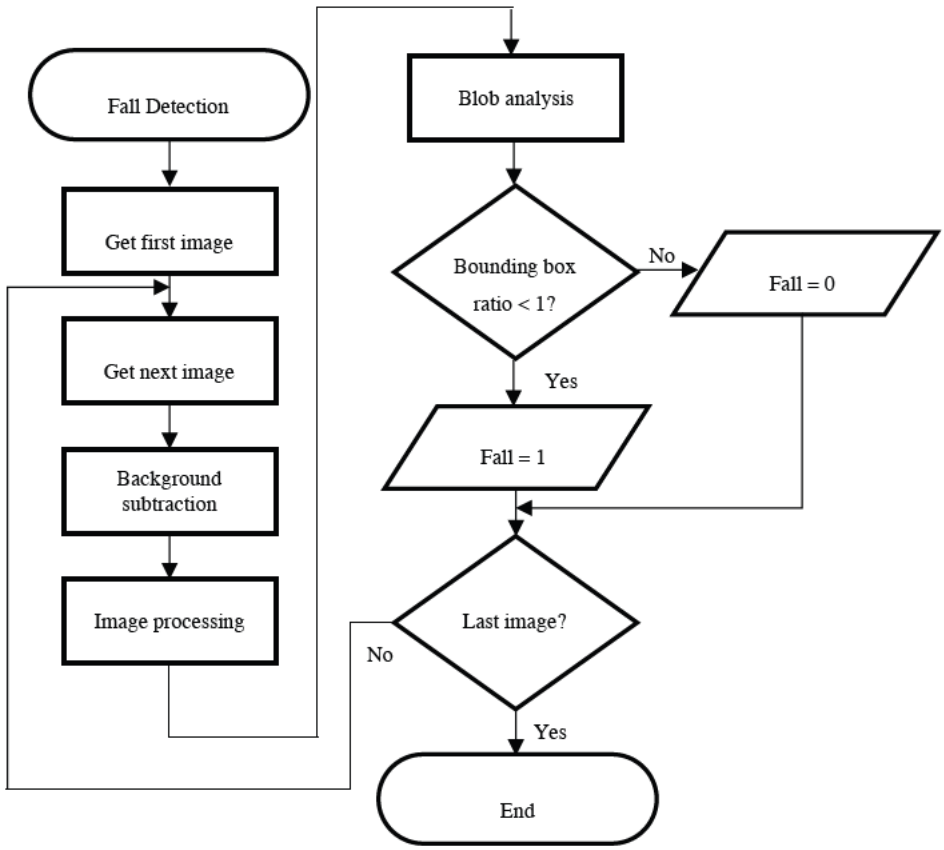

Figure 4.4 Flow chart for bounding box fall detection method

4.1.3 Segmentation using MATLAB foreground detector

It can be seen above that when image segmentation is poor, the fall detection algorithm has a difficult time correctly determining a fall. For the success of this project a more reliable method of image segmentation was sought. In its computer vision toolbox, MATLAB has a function, vision.ForegroundDetector, which is a foreground detector that uses Gaussian Mixture Models (GMM) to return a foreground mask and is derived from the work by Kaewtrakulpong and Bowden (2001) and Stauffer and Grimson (2007). The foreground detector has several parameters that can be set to make it work effectively. Adapt learning rate allows the detector to update the learning rate with each frame, the learning rate is how rapidly the algorithm adjusts to changing conditions. The number of training frames can also be adjusted and needs to be set to an appropriate number so the algorithm can get a robust background image. The minimum background ratio determines what pixels are to be considered background values and must be appropriately set to allow for multimodal backgrounds. There is also the ability to set the number of gaussian modes as well as the initial variance of the model.

Used in conjunction with the foreground detector is another function from the computer vision toolbox, vision.BlobAnalysis. This function analyses all the connected regions, or blobs, in the binary image and produces statistical outputs that can be used to describe the blobs. Whilst there are plenty of outputs that can be used for computer vision projects, the properties used for this project were centroid, area, bounding box, major axis length, minor axis length and orientation output. The centroid output returns the coordinates on the image of the centre point of the blob and was used to determine the distance and therefore the speed, at which the blob was moving between image frames. The area output port was used to filter any smaller blobs in the image that would not be large enough to represent a human and as per the initial fall detection algorithm the bounding box was used to represent the region of interest (ROI) for the blob. The major and minor axes outputs are used to represent the ellipsis that defines the blob, and the orientation is used to represent the angle of the major axis in relation to the x axis of the image.

To produce a foreground mask the foreground detector needs to be trained to detect the background model. This meant that the datasets needed an image sequence at the start that contains just the background with no human or other moving objects. This narrowed the available datasets even further than the original segmentation method as the background subtraction method could rely on the one reference image that was supplied with some of the datasets. The foreground detector relies on a number of training frames to be effective and through trial and error the optimum number has been set at around 40. If this number is set too small the foreground detector will detect minor lighting changes and shadows which would make the segmentation unstable. The number of training frames could be set higher, however, the datasets would run out of available images. Like the number of training frames, the learning rate was set with a fair amount of trial and error. If the learning rate is set too high, the algorithm will quickly erode the foreground mask which creates problems if the person in the frame is stationary for an extended period. If it is set too low, then any minor change in the background image will remain in the frame causing nuisance blobs. The minimum

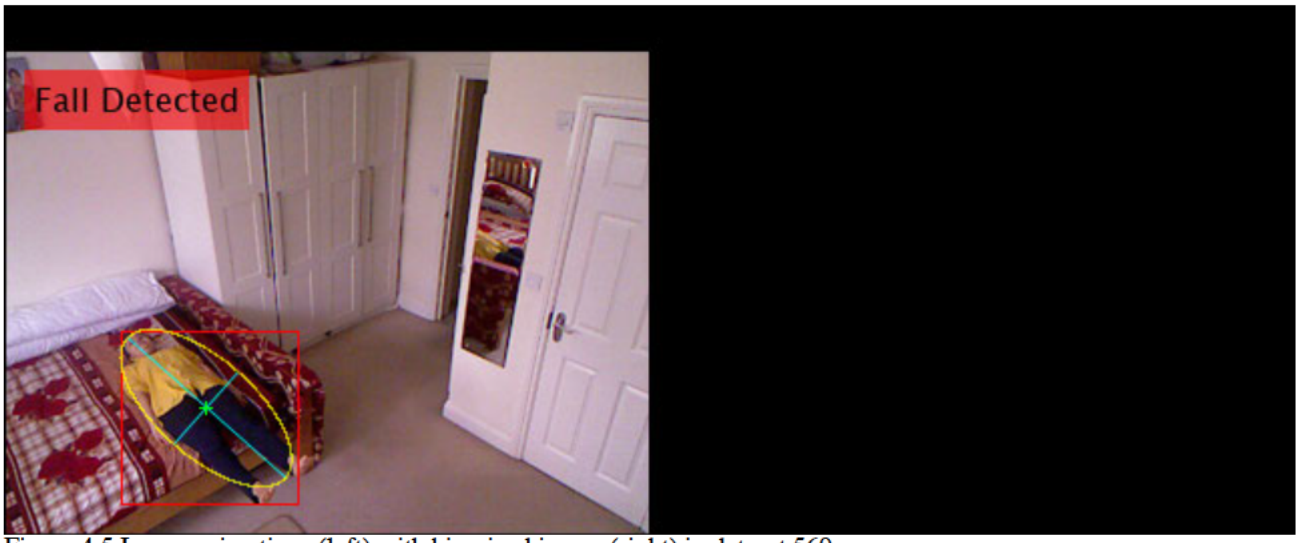

Figure 4.5 Image animations (left) with binarized image (right) in dataset 569

background ratio was generally left around the default value of 0.7 for most of the datasets. This parameter sets the minimum possibility that the algorithm will consider for pixels to be deemed background pixels and when set too low the foreground mask would be smaller and sometimes undetected, especially in darker areas of the image. Finally, the initial variance was generally set to auto. This was found to produce the best results throughout most datasets.

As with the original image processing method, the program would read the individual image from the image datastore. In this case, however, the foreground detector would produce the binarized image prior to being opened and filled before image morphology is completed. The process changes slightly after this where the outputs from the blob analysis are put into arrays so they can be utilised by the fall detection algorithm and other processes.

Whilst not critical for the success of the algorithm, it is important that there is a visual representation of the blob analysis outputs so that the person assessing the algorithm can see what is happening in real time. For this reason, several animations have been included that will overlay on each image as it is read by the program. Firstly, the major and minor axes are shown to illustrate how the detector is representing the angle of the human as well as the height and width. Secondly, an ellipse has been drawn around the blob using the major and minor axes outputs. The blob analyser does not produce an ellipse output, therefore the ellipse was produced using the following equation (Cookie Robotics n.d.).

$$\begin{bmatrix} x \\ y \end{bmatrix} \quad \begin{bmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} a\cos t \\ b\sin t \end{bmatrix} \qquad \text{Equation 4-1}$$

Where, x and y are the coordinates for each point on the ellipse, φ represents the angle of the major axis in relation to the x axis of the image, a and b are the lengths of the major and minor axes respectively and t is a series of equally spaced values from 0 to 2π. What gets returned is a matrix with all the x and y coordinates to draw an ellipse around the blob on the image. A bounding box around the object is also produced, purely to highlight the image frames when a fall occurs. Colour changing text is also produced in the upper left corner

of the image to show the viewer when a fall has been detected. An example of the annotated image is shown on the left side of Figure 4.5 and on the right is the segmented image.

Table 4.1 Dataset 569 results example (extract)

Image No.	Image Label	Detector Result	Compare Result
122	2	0	TN
123	2	0	TN
124	3	0	FN
125	3	1	TP
126	3	0	FN
127	3	0	FN
128	3	0	FN
129	3	0	FN
130	3	0	FN
131	3	0	FN
132	3	1	TP
133	3	1	TP

The final part of the main program takes the results from the fall detection algorithm and puts them into a column on the label spreadsheet with each cell lined up with the corresponding image number. From there simple excel formulas were used to compare the result with the image label and resolve if the fall detector had correctly detected a transition from standing to falling. Table 4.1 shows an extract from the results for dataset 569 that shows the image number, the label given by the dataset, the fall detection result and then the comparison result.

Adhikari et al. (2017b) developed their datasets mainly for use in deep learning applications using convolutional neural networks (CNNs) where they were trying to train the computer for various poses, so the following labels were included. 1 is standing, 2 is sitting, 3 is lying, 4 is bending, 5 is crawling and 0 is empty. For the purposes of this project, the algorithm is only concerned if the person is in the lying pose. Therefore, if the image label was 3 and detector result was 1, then a true positive (TP) is detected and if the image label is not 3 and the detector result was 0, then a true negative (TN) result was detected and so on.

Most of the tuning and debugging of the image segmentation program and the fall detection algorithm were conducted using dataset 569. This dataset had reasonable lighting with few dark spots in the image and the position of the camera was far enough back to give a wide enough view of the room. The only real challenges were the bed creating an occlusion when the subject stood behind it and a mirror on a wall which reflected the image of the person when they walked near it. Several iterations of the foreground detector segmentation

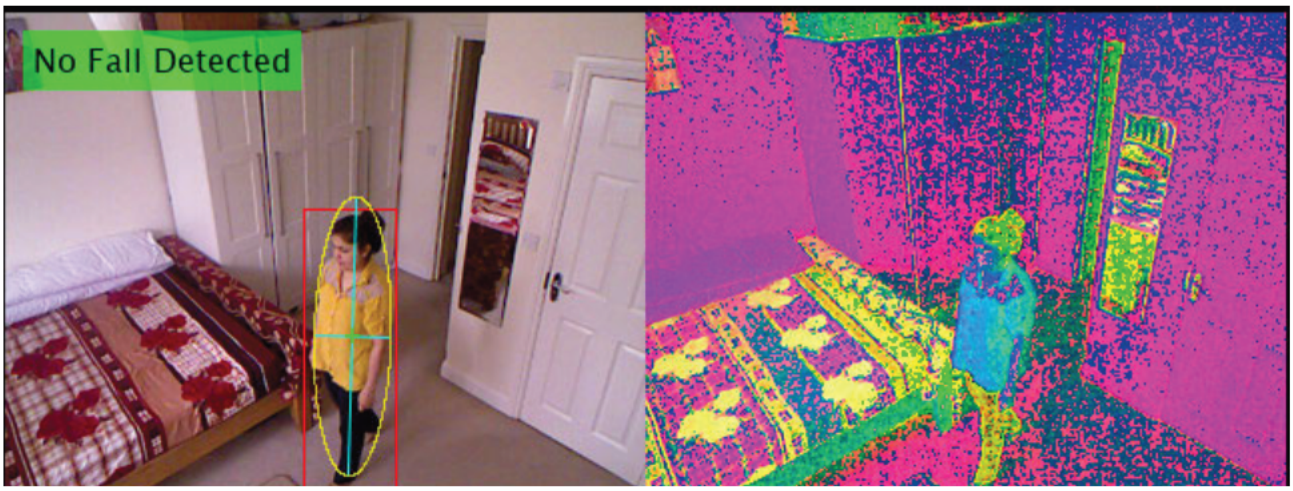

Figure 4.6 RGB image converted to HSV

method were developed throughout the life of this project to optimise the foreground mask. Early on it was discovered that shadows cast by the person were being detected by the foreground detector and distorting the foreground mask. A MathWorks forum suggested that the foreground detector be used on a hue, saturation, value (HSV) image (MathWorks 2016). An HSV image created from an RGB image by MATLAB will produce an *m*-by-*n*-by-3 matrix where the third dimension is the hue, saturation and value figures for each pixel in the image. The hue is a value from 0 to 1 that corresponds to a value on a colour wheel, saturation is the level of hue value and value is the maximum value of the red, blue and green components of a specific colour (MathWorks 2021c). The result of this change successfully eliminated the effect of shadows and image segmentation was vastly improved. Figure 4.6 shows the resultant image of the RGB to HSV conversion and the reduction in shadow casting is noticeable.

Figure 4.7 Binary image with no morphological operations (left) and after dilation and erosion (right)

Morphological operations to the binary image affect the segmentation by manipulating the size and shape of the blob. Applying a dilating function to the image has the effect of opening the foreground mask and filling any gaps between sections of the blob. Then, by applying an eroding function, the width of the blob is trimmed to a size that should reflect the size of the person in the image. The left side of Figure 4.7 shows the foreground mask without any morphological operations and on the right is the mask after the morphological operations have been applied.

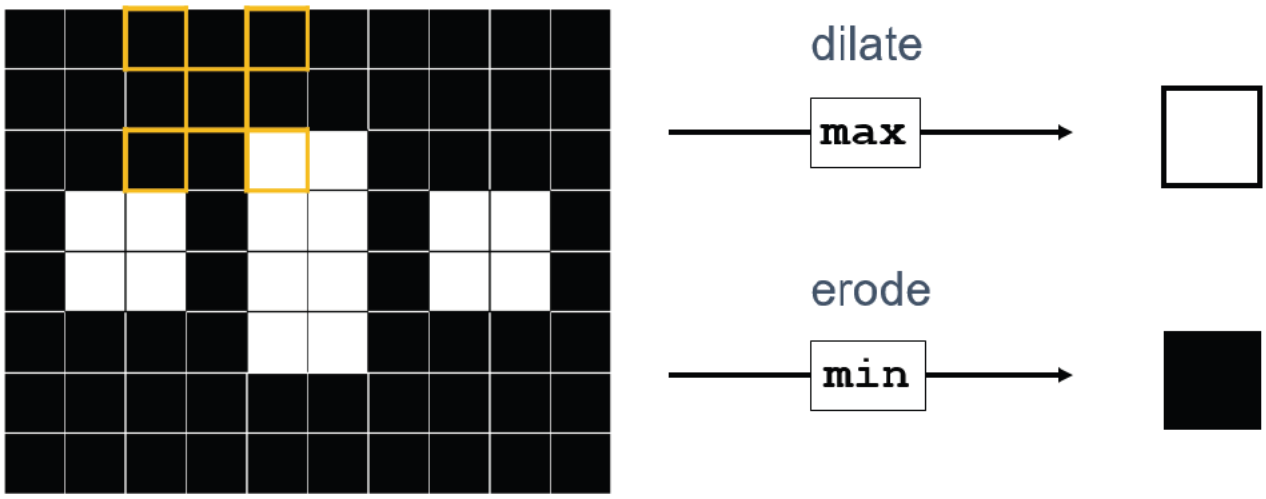

Figure 4.8 Morphological operations principle of operation (MathWorks 2021a)

By applying the morphological operations any separation of the segmentation will be ignored and the shape will stay in one piece and give one solid blob which represents the human shape. Morphological operations use a specified shape that is applied to the binary image and filters the target pixel based on the value of the neighbouring pixels. In a dilation operation the target pixel will be set based on the largest value pixel in the neighbouring pixels, whilst the erosion will be looking for the minimum valve pixel (MathWorks 2021b). Figure 4.8 shows the principle of operation. For this project, dilating and eroding the foreground mask improved the performance of the fall detection algorithm and was used in the final program.

4.1.4 Fall detection using blob bounding box, centroid distance, axis ratio and angle

The next generation of fall detection algorithm adopted a method similar to Rougier et al. (2007) who utilised the motion history image and the standard deviations of the orientation angle and the ellipse axis ratios. Instead of using a motion history image, this algorithm uses the distance the centroid travelled between each frame. The theory behind using this parameter is that during a fall event the blob should show a greater displacement between frames and by setting an appropriate threshold the algorithm can detect the fall. To determine the distance the centroid travels between each frame, the following formula was used,

$$d_i \quad \sqrt{(x_i \quad x_i \quad)^2 + (y_i \quad y_i \quad)^2} \qquad \text{Equation 4-2}$$

Where d_i is the centroid distance change from the previous image to the current image, x_i and y_i are the x and y coordinates of the blob centroid in the current image while x_{i-1} and y_{i-1} are the x and y coordinates of the blob centroid in the previous image. Figure 4.9 shows how the calculation for centroid distance is derived.

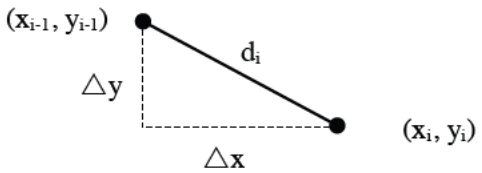

Figure 4.9 Diagram defining the centroid distance calculation

The objective of the orientation angle is to capture falls when the person is falling in a direction parallel with the horizontal axis of the camera. That is, when the person falls the angle should change from somewhere around 90° to an angle towards 0° or 180°, depending on the direction of fall. Axis ratio change, on the other hand, is used to detect those falls which occur parallel with the vertical axis of the image. It is expected that with these types of falls there would be not a lot of change in orientation angle. Standard deviation has been used with these parameters to allow a certain level of variability in the object features. A sudden change in the level of either of the parameters should increase the standard deviation and with correct setting of the thresholds, falls should be easily detectable. The standard deviation of these parameters is calculated as follows,

$$\sigma_{\varphi} \quad \sqrt{\frac{\sum(\varphi_i \quad \varphi)^2}{N}} \qquad \text{Equation 4-3}$$

$$\sigma_{\rho} \quad \sqrt{\frac{\sum(\rho_i \quad \rho)^2}{N}} \qquad \text{Equation 4-4}$$

Where, φ_i is the orientation angle in radians of the blob in the ith image, $\bar{\varphi}$ is the mean of the angles in the chosen dataset, N is the number of angles and axis ratios in the dataset and σ_{φ} is the standard deviation of the angles in the selected dataset. ρ_i is the axis ratio of the blob ellipse in the ith image and $\bar{\rho}$ is the mean axis ratio and σ_{ρ} is the standard deviation of the axis ratios in the dataset. For these parameters, N has been set at 5. It was found that this number of images is enough to accurately detect a change in posture and not too large that it might take longer for the variation to take effect.

Table 4.2 Image centroid distance, axis ratio and angle standard deviation data around a fall at image 124

Image No.	Label	Fall	Centroid Distance	Axis std dev	Angle std dev
120	2	0	1.560357148	0.306393007	0.109383081
121	2	0	1.163405087	0.301346134	0.116292476
122	2	0	2.434893413	0.266909075	0.114863866
123	2	0	9.424564613	0.232503059	0.084728255
124	3	0	5.760408859	0.272827704	0.098549742
125	3	1	6.059147855	0.41687095	0.123030603
126	3	0	5.53645562	0.495472985	0.141785674
127	3	0	2.472377735	0.575319511	0.124478138
128	3	0	1.714939255	0.54991545	0.132643618
129	3	0	1.535566164	0.271602316	0.169630865
130	3	0	3.64241218	0.225893382	0.17500733

Setting the most appropriate thresholds for these parameters was a lot more complex and involved analysing the standard deviations of the angles and axis ratios around the fall images. Using dataset 569, the centroid distance and the standard deviation data from the orientation angle and the axis ratio in each image were included in the output spreadsheet. This allowed easy analysis of the parameters by cross referencing the images where a fall had occurred. For instance, in dataset 569 a lying pose occurs in images 124 to 150. By analysing the centroid distance and the standards deviations of the angle and axis ratios around this point, some idea of thresholds can start to be determined. Table 4.2 shows the centroid distance and standard deviation data around image 124 where the fall begins to occur. It can be seen that after image 124 there is a noticeable increase in the standard deviation of the axis ratio and a smaller increase in the angle standard deviation. There is also a noticeable increase in the centroid distance at image 123 that could show the person has started to transition from standing to lying and that a fall is taking place.

The next lying pose in this dataset occurs between images 187 and 218. Table 4.3 shows the results of the axis ratio and angle standard deviations around image 187. Once again, there is a noticeable change in axis ratio although on this occasion it is just before the change in pose. The angle standard deviation also changes slightly just after the change in pose. Figure 4.10 shows the image sequence from image 180 to 191 including the associated binary image. As shown in Table 4.3, the axis ratio standard deviation gradually increases and peaks at around image 185. This is due to the lengthening of the blob as the subject person moves from the standing pose to the lying pose. The standard deviation of the angle increases slightly at image 190 where the person makes contact with the bed. Centroid distance also has an increase at this image, however, there are similar sized increases in the images preceding the change in pose which are probably due to the subject moving on to the bed in a kneeling pose prior to the lying pose.

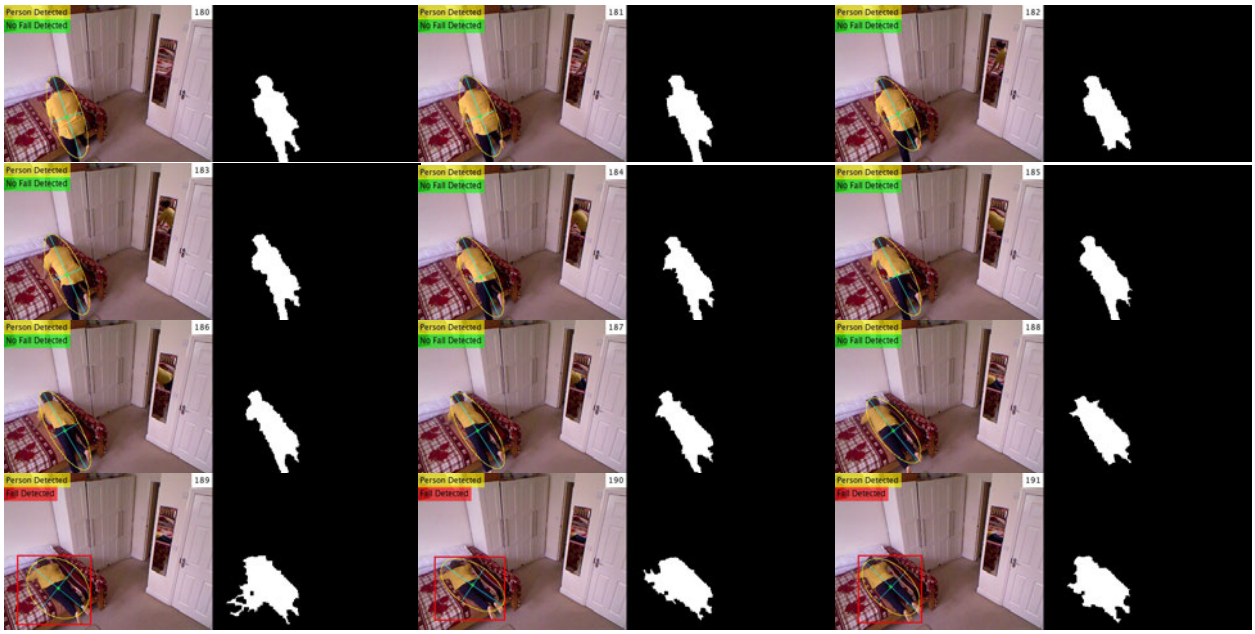

Figure 4.10 Image sequence (180 to 191) showing axis ratio and orientation angle change in dataset 569.

Table 4.3 Image centroid distance, axis ratio and angle standard deviation data around a fall at image 187

Image No.	Label	Fall	Centroid Distance	Axis std dev	Angle std dev
180	1	0	8.218887195	0.339565326	0.075092829
181	1	0	6.784447853	0.362334411	0.078679621
182	5	0	2.103510673	0.351430695	0.085140495
183	5	0	3.01666616	0.353659541	0.082668536
184	5	0	7.239624149	0.416287117	0.08727426
185	5	0	1.043243411	0.441055839	0.017690059
186	5	0	9.367018826	0.350925332	0.028143606
187	3	0	4.158301809	0.264510263	0.03519436
188	3	0	5.620140545	0.143986444	0.044816998
189	3	0	2.638985479	0.28882949	0.093323026
190	3	1	3.41988733	0.314117238	0.120453008
191	3	1	0.953639188	0.308431589	0.110289501

Table 4.4 Image centroid distance, axis ratio and angle standard deviation data around a fall at image 423

Image No.	Label	Fall	Centroid Distance	Axis std dev	Angle std dev
413	5	0	2.311564903	0.076981551	0.675617847
414	5	0	1.673144656	0.124675243	0.648456155
415	5	0	2.425606443	0.130589467	0.295274149
416	5	0	1.050091411	0.124613957	0.302941545
417	5	0	1.666258154	0.101574546	0.245005844
418	5	0	6.748841473	0.080753686	0.161068901
419	5	0	1.181243937	0.050675003	0.08807488
420	5	0	3.587555701	0.048753522	0.071793481
421	5	0	5.173046718	0.082081626	0.064667929
422	5	0	5.524813121	0.163266677	0.067040354
423	3	0	2.948186969	0.239744735	0.068164287
424	3	1	5.460879161	0.370691563	0.057088843
425	3	1	0.901023463	0.364284319	0.061793704
426	3	1	1.749388901	0.32675667	0.038202394
427	3	0	1.330919809	0.202816471	0.030648461
428	3	0	1.108660239	0.198847495	0.033661214
429	3	0	26.9791442	0.453914223	0.084306955
430	3	0	1.79097219	0.467129323	0.093045996

There is a third and final event in dataset 569 that can be used to train the fall detector at image 423 where the subject goes into a lying posed that lasts until image 518. Table 4.4 shows the centroid distance and standard deviation results around the start of the lying pose and once again there is a slight increase in axis ratio standard deviation several frames after the start of the lying pose label at image 429. What is also notable is that there is very minimal variation in orientation angle at the time of the change in pose. This is due to the manner in which the subject moves to the lying pose and as such, the variation in angle comes before the movement to the lying pose in image 418. The centroid distance also shows a change in magnitude just before the lying pose also at image 418. Figure 4.11 shows the change in orientation angle prior to the lying pose from image sequence 408 to 416 and the continuation of this sequence shows the movement from standing to lying finishing in image 424.

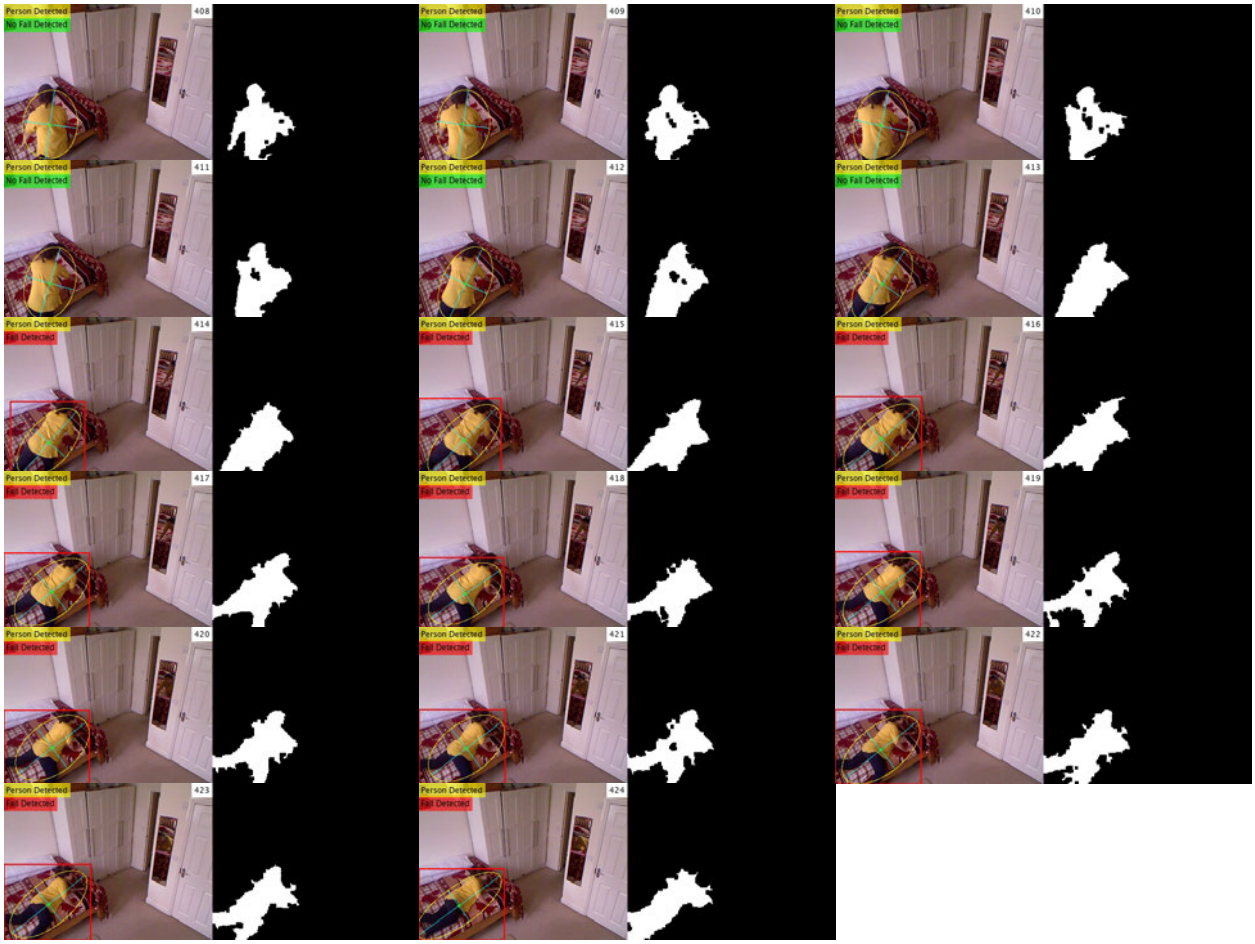

Figure 4.11 Image sequence 416 to 424 showing standing to lying pose

The results of the centroid distance, axis ratio and orientation angle standard deviations show that it is possible to create a fall detection algorithm using these parameters. However, the variability of each parameter means that there are instances when the parameter might reach the threshold outside of a change from standing to lying pose. For this reason, the algorithm was first designed so that a possible fall could only be detected when at least two of the parameters had reached their threshold. That is, a possible fall is detected under the following scenarios.

- if the centroid distance and the standard deviation of the axis ratio exceed their thresholds
- if the centroid distance and the standard deviation of the angle exceed their thresholds
- if the centroid distance and the bounding box ratio exceed their thresholds
- if the axis ratio and angle standard deviations exceed their thresholds
- if the bounding box ratio and the standard deviation of the angle exceed their thresholds
- if the bounding box ratio and the standard deviation of the axis ratio exceed their thresholds

This algorithm was run using dataset 569 using the following thresholds with the results shown in Table 4.5.

- Centroid distance – 5 pixels
- Axis ratio standard deviation – 0.25
- Angle standard deviation – 0.5

Figure 4.12 shows the flow diagram for the algorithm where a fall detected flag is returned to the main program if any of the above conditions are met.

Table 4.5 Results of initial fall algorithm on dataset 569

Dataset 569	
TP	36
TN	404
FP	11
FN	119
Accuracy	77.2%
Sensitivity	23.2%
Specificity	97.3%

It would appear from these results that the algorithm has good accuracy, however, when viewed in conjunction with the sensitivity it shows that it was not good at detecting a maintained lying pose. What was encouraging though, was that when viewing the image sequence while running the fall detector over the dataset, it regularly detected the change in pose from standing to lying. The problem was that the algorithm could not tell if the person remained in the lying pose due to only small changes in axis ratio, angle, centroid distance and bounding box ratio. This is evident in Table 4.2 and Table 4.4 where in the third column a fall flag has been added approximately around the images that show the change in pose before changing back to no fall despite the image being labelled a lying pose. This initial fall algorithm was then run on other datasets that could be used with the MATLAB foreground detector and the results are shown in table 4.6. As was the case with dataset 569 this algorithm shows poor sensitivity predominantly due to the algorithm being unable to detect if a person has remained in the lying pose.

There are several key observations that were taken away from this version of algorithm that would help in future revisions. Firstly, the algorithm consistently detected a change in pose from standing to lying, however, it was unable to maintain the fall detection if the person remained in the lying pose and movement was minimal.

This characteristic will need to be addressed in future revisions as it is critical that the algorithm is able to distinguish a person who has stayed fallen so it can prevent the 'long-lie'. Secondly, there is a perceptible difference in detection when the rate at which the person falls is faster. This could help distinguish between actual falls and those where the person is just lying down. Also, with this version of fall detector HSV image conversion was not conducted, so the foreground mask was affected by shadows which in turn affected all the parameters utilised by the fall detector. Future versions of the detector would use the HSV image conversion as detailed in section 4.1.3 above. It was also discovered when testing this algorithm that image segmentation was affected by several factors including lighting variations in the image, for example, sunlight shining onto the floor from an uncovered window. There were also some datasets that had dark areas in which the foreground detector would struggle to find the person and others where the colour of clothing would cause low contrast which would break or shorten the blob. These faults obviously affect the performance of the fall detection algorithm and have mainly been addressed in later versions by manipulating some of the parameters of the MATLAB foreground detector function.

Table 4.6 Results of initial fall algorithm on remaining datasets

	Dataset 758	Dataset 786	Dataset 832	Dataset 1954	Dataset 2123
TP	57	48	54	182	90
TN	649	587	498	1457	1354
FP	33	12	48	114	74
FN	19	139	232	201	482
Accuracy	93.1%	80.8%	66.3%	83.9%	72.2%
Sensitivity	75.0%	25.7%	18.9%	47.5%	15.7%
Specificity	95.2%	98.0%	91.2%	92.7%	94.8%

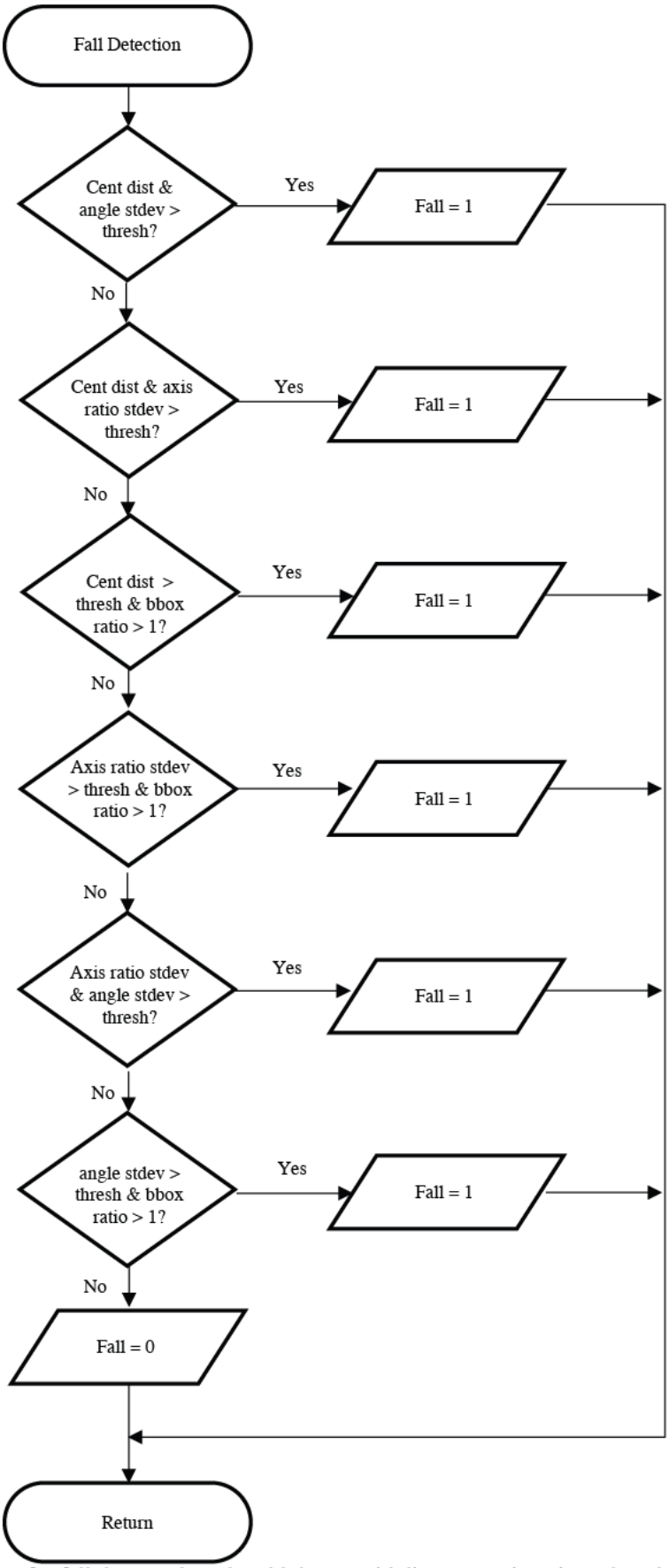

Figure 4.12 Flow diagram for fall detector based on blob centroid distance, axis ratio and angle

4.1.5 Revised fall detection using HSV image conversion

The next iteration of fall detection algorithm still utilises the MATLAB foreground detector with the blob analyser, however the number of training frames was increased to the maximum so a more solid background could be achieved, and the minimum background ratio was altered so that the foreground mask was not affected so much by lighting changes. Also, the images have been converted to HSV from the standard RGB. This was done to alleviate the issues caused by shadowing and in turn hopefully improve the performance of the fall detector. Figure 4.13 shows an example of the changes to the binary image when HSV images are used. In the left image the shadow has changed the shape of the foreground mask which has resulted in a wider ellipse. On the right is the same image number, but the image was converted to HSV prior to running the foreground detector. The improvement in the binary mask is obvious.

Figure 4.13 Effect of converting to HSV for shadow removal in dataset 569

Table 4.7 shows the results of the revised image processing method on the centroid distances, axis ratio and angle standard deviations for the change on pose around image 124. When compared with Table 4.2, the figures appear more consistent, particularly the centroid distance, with the new image processing technique.

Table 4.7 Image data around a fall at image 124 using revised fall detector

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev
120	2	0	2.539114184	0.564516129	0.235736408	0.089066824
121	2	0	1.459800299	0.624	0.261903668	0.09135033
122	2	0	4.352537382	0.753968254	0.293049089	0.105694117
123	2	0	3.815235982	0.725806452	0.194682603	0.096370515
124	3	0	4.433299493	0.794642857	0.194859393	0.10270665
125	3	1	6.276742827	1	0.322297111	0.131145398
126	3	0	4.033276784	0.904761905	0.490147203	0.140136706
127	3	0	1.736443738	0.938271605	0.638479927	0.120004729
128	3	0	1.935286942	0.824175824	0.610359181	0.132043009
129	3	0	0.571761309	0.802083333	0.461580446	0.184036721
130	3	0	1.79525915	0.941176471	0.30719427	0.192311318
131	3	0	4.746498997	1.194029851	0.151337972	0.269320973
132	3	1	2.136955762	1.428571429	0.263439608	0.315775938
133	3	1	8.370122382	1.148148148	0.38774422	0.319619722

The fall algorithm has also been revised to help detect if a person has stayed fallen. So, a fall is detected when,

- The bounding box average ratio from the previous five frames is greater than one.
- The standard deviation of the angle of the of the blob in the previous five frames is greater that the threshold AND the standard deviation of the axis ratio from the previous five frames is greater than the threshold.
- The centroid distance from the previous frame to the current frame is greater than the threshold AND the standard deviation of the angle of the of the blob in the previous five frames is greater that the threshold.
- The centroid distance from the previous frame to the current frame is greater than the threshold AND the standard deviation of the axis ratio from the previous five frames is greater than the threshold.

The fall column in Table 4.7 shows that while the detector has picked up the initial change in pose at image 125, there is still a period where the person is in the lying pose and the detector has given a non-fall state. The fall flag is then picked up again at image 132 and remains on for most of the remainder of the period when the person is in the lying pose. This is due to the way the person falls on to the bed. The image sequence displayed in figure 4.14 shows that after the person lands on the bed, their legs go up into the air which keeps the centroid relatively still, the bounding box ratio doesn't increase above threshold and the angle doesn't vary enough to trigger the threshold.

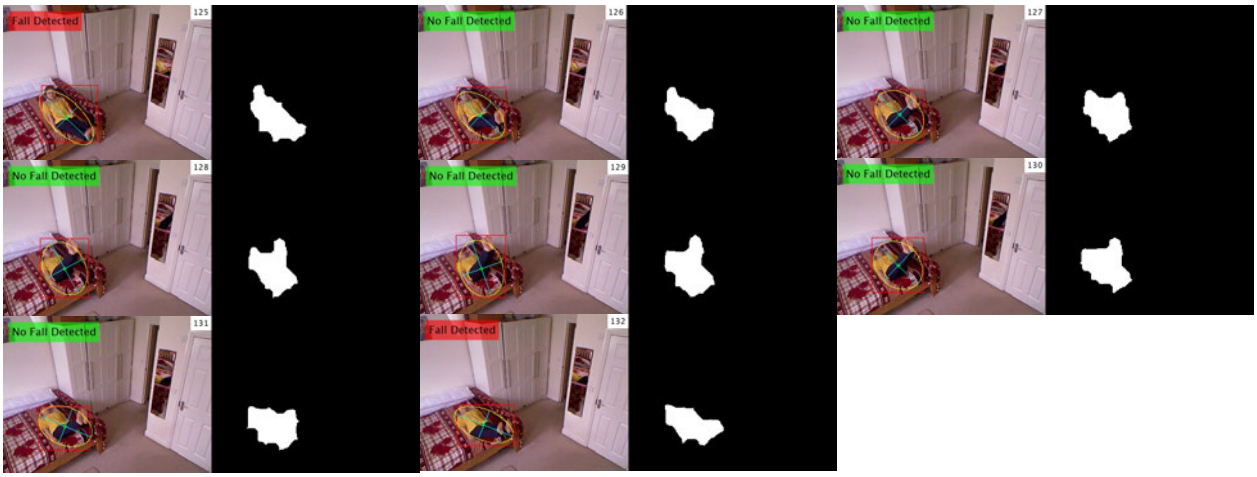

Figure 4.14 Image sequence (125 to 132) showing a missed fall due to an unusual lying pose in dataset 569

In the next fall sequence, the detector has picked up the change to a lying pose within one frame of the image label, see Table 4.8, and alternates between on and off for the duration of this lying pose. After the person stands up, however, the detector continues to incorrectly show a fall state for several image frames. This is due to the foreground detector yielding an abnormal blob and distorting the ellipse and the bounding box as the person gets up off the bed. It is unclear as to why the foreground detector has given this blob shape, however, in Figure 4.15 it can be seen that it is likely that the detector has latched onto the movement of the stripe on the sheet on the bed from the person lying on top of it. The blob appears to hold on to the stripe until the person gets up off the bed.

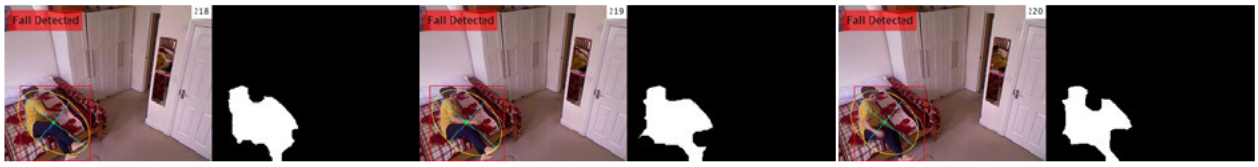

Figure 4.15 Foreground detection anomaly causing distorted ellipse and bounding box in dataset 569

The final change to lying pose in this dataset has been not only well detected by the fall detection algorithm, but also easily maintained as the person remained in the lying position. As shown in Figure 4.16, this is mainly due to the lying pose being a classical type where the person lies parallel with the horizontal axis of the camera so was easily detected and retained by the bounding box ratio parameter.

Table 4.8 Image data around a fall at image 187 using revised fall detector

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev
184	5	0	0.864669149	0.603053435	0.141059103	0.069258427
185	5	0	1.768559446	0.621212121	0.044559543	0.046570321
186	5	0	0.948221501	0.62406015	0.092045758	0.045743583
187	3	0	2.599446257	0.661654135	0.082279605	0.051929381
188	3	1	5.977087777	1.028301887	0.216565707	0.10450184
189	3	1	12.40263839	1.066666667	0.618118655	0.225571926
190	3	1	6.853161192	1.208333333	0.637810827	0.236017701
191	3	0	2.571406947	1.058823529	0.615901415	0.213014124
192	3	1	5.299633416	0.922330097	0.498925891	0.177829621
193	3	1	4.201669126	1.102040816	0.339249531	0.127756741
194	3	1	3.09942585	1.082474227	0.284524671	0.11330993
195	3	1	3.157410386	0.871287129	0.124114494	0.081480531

Table 4.9 Results of revised algorithm on dataset 569

Dataset 569	
TP	126
TN	330
FP	85
FN	29
Accuracy	80.0%
Sensitivity	81.3%
Specificity	79.5%

Overall, the revised fall detection algorithm has vastly improved on the previous fall detection algorithms. The results of the testing conducted on dataset 569 shown in Table 4.9 will back that statement up, when compared

with the results in Table 4.2. However, there is still much improvement to be made. While the sensitivity has increased, meaning the algorithm is detecting more lying poses, it has come at the cost of specificity, which means that there are a greater amount of false positives. The goal of the succeeding algorithm revisions will be to analyse the images that are showing incorrect states and adapt the fall detection algorithm to ignore the features that are causing them.

Figure 4.16 Fall detection of near horizontal lying pose

4.1.6 State-based fall detection with upright angle reset

The previous versions of fall detection algorithms using centroid distance, axis ratio and angle standard deviations were both able to accurately detect a change in pose to within a few frames of the image label state. However, both have struggled to accurately maintain the fall status after the person has fallen and before they stand up. In this revision of algorithm, a state-based control will be used where after a fall is detected the state will change to a fall condition and will remain in that state until the algorithm has detected that the person is upright again. Figure 4.17 shows the state diagram and which conditions will trigger the change of state.

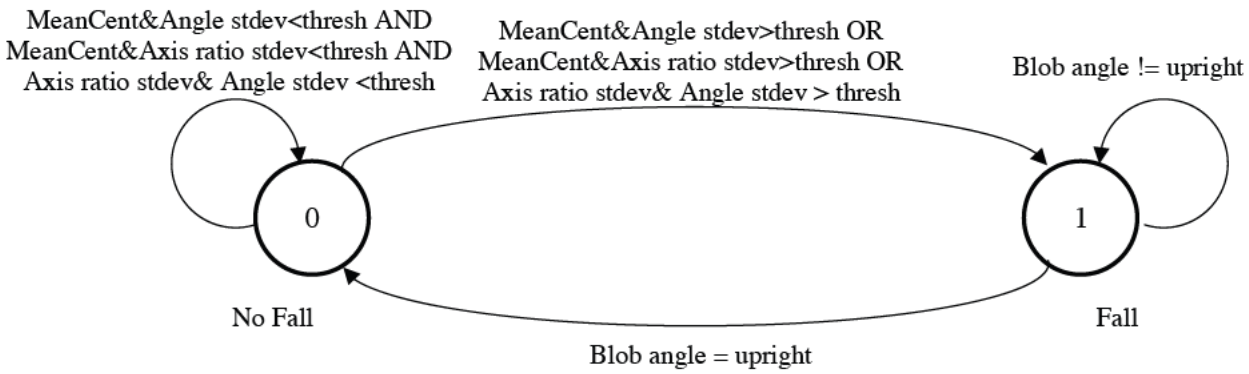

Figure 4.17 State diagram of fall detection algorithm

To ascertain if a person is upright, the algorithm uses thresholds in the orientation angle. If the angle is outside these thresholds on either side, then the person is deemed to be not upright. To determine the most appropriate upright angle thresholds, the orientation angle was included in the image data spreadsheet and analysed. Table 4.10 shows the image data from the first fall event in dataset 569 and includes the orientation angle in radians.

If the person was completely upright, then it would be expected that the orientation angle of the ellipse would be around 1.57 radians, or 90°. In Table 4.10, the orientation angle at the time of the fall in image 124 is around 2.2 radians (126°). Of greater interest is the orientation angle after the person stands up following the lying pose. This data will help determine the angle at which to reset the fall detector back to the no fall state. Table 4.11 shows the image data after the person sits up after a fall with the image where the label changes from lying to sitting highlighted in green. The angle at the time of change in state from lying is similar to that seen in Table 4.11 at time of change of state to lying of 2.1 radians. After the third lying event in dataset 569 between images 518 and 519, the orientation angle is much smaller than the first lying pose event, however, it is of approximately equal angle from the vertical. This angle is approximately 0.53 radians or about 30°, therefore, the thresholds for the upright angle have been set at 1.04 radians (60°) to 2.1 radians (120°).

Table 4.10 Image data from dataset 569 showing orientation angle at time of lying pose

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev	Orientation angle
120	2	0	1.318202773	1.964845576	0.183403298	0.068789236	1.967214989
121	2	0	1.052677671	1.656692282	0.205277701	0.068888816	2.001754183
122	2	0	2.135745492	1.502208646	0.197160032	0.075290364	2.090794567
123	2	0	3.051797195	2.080073453	0.163410376	0.081692263	2.136621181
124	3	0	3.763233589	2.983592092	0.125399688	0.090384589	2.216807643
125	3	1	5.640350822	4.151793869	0.309745453	0.119570063	2.35462853
126	3	1	4.641853197	4.681812536	0.523860152	0.14360782	2.480295362
127	3	1	1.530672647	3.937625555	0.654317192	0.140699048	2.490448997
128	3	1	2.909496506	3.027340783	0.595943307	0.149800452	2.105388315
129	3	1	2.037908663	2.159359272	0.423070644	0.241299184	1.864228116
130	3	1	0.813511876	1.920305681	0.215175041	0.294949115	2.675657944

Table 4.11 Image data from dataset 569 showing orientation angle after person gets up after lying

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev	Orientation angle
147	3	1	3.839181583	2.646102045	0.042515763	0.028722439	2.355057078
148	3	1	2.720373031	2.653706973	0.04813899	0.028302987	2.36146833
149	3	1	1.403397962	2.654317525	0.072337362	0.038289064	2.276551314
150	3	1	0.524407886	1.54939296	0.077845731	0.060282327	2.221540115
151	2	1	1.235584235	1.054463361	0.078181135	0.08415225	2.138026665
152	2	1	1.775178828	1.178390316	0.065058148	0.094033783	2.100504595
153	2	1	4.269687598	2.426816887	0.123267503	0.069035304	2.106943084
154	2	1	1.989523423	2.67812995	0.155348206	0.046769297	2.096424192
155	2	0	2.464971419	2.908060813	0.154414186	0.033837361	2.034182168

Table 4.12 Image data showing orientation angle in third lying pose in dataset 569

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev	Orientation angle
514	3	1	3.313297308	2.738902869	0.393496303	0.011340561	0.54409155
515	3	1	7.328885415	4.434834802	0.441349657	0.012593345	0.53421047
516	3	1	8.826271427	6.489484717	0.506199278	0.06048659	0.700465265
517	3	1	3.069555039	6.408237294	0.331168051	0.117260425	0.836965409
518	3	1	3.057178466	4.984334977	0.221490622	0.193098134	1.048220036
519	2	1	1.066552343	2.397761949	0.24164003	0.21563661	1.116436747
520	2	1	2.188286899	2.104005903	0.168569462	0.168481219	1.125314111
521	2	1	1.464091272	1.572976838	0.223256315	0.119532643	1.178796469
522	2	1	3.168278342	2.273552171	0.221006308	0.045074057	1.161060558
523	2	1	5.342035365	3.32480166	0.151455184	0.077805406	0.959439891

Knowing the angle thresholds for upright orientation, a fall is detected when the following conditions exist.

- The average centroid distance of the previous three frames is greater than the threshold AND the angle standard deviation of the previous five frames is greater than the threshold AND the orientation angle is not in the upright limits.
- The average centroid distance of the previous three frames is greater than the threshold AND the axis ratio standard deviation of the previous five frames is greater than the threshold.
- The standard deviation of the angle of the of the blob in the previous five frames is greater than the threshold AND the standard deviation of the axis ratio from the previous five frames is greater than the threshold.

The centroid distance was averaged to smooth out some of the random spikes and create a more representative distance travelled over time. The first criteria, which looks at the centroid distance with the angle standard deviations is also dependent on the orientation angle being outside of upright limits. This means that a detected fall due to centroid distance and orientation angle standard deviation will be ignored unless the orientation angle is outside of the upright limits. This had the effect of removing several nuisance false positive tags from the dataset. This condition was not added to the other two criteria because it may mask a change in pose to lying that is parallel with the vertical axis of the camera which can only be detected by the axis ratio standard deviation. The bounding box ratio criteria was removed from this algorithm as it was found to provide no additional benefit to fall detection that the other criteria were already detecting and was, in fact, the cause of several false positives.

This version of fall detection performed better in all three measures of performance as shown in Table 4.13. Increasing the specificity to over 97% means that the algorithm is unlikely to miss a fall event. There are,

however, still quite a few false positives which means that the algorithm is picking up anomalies with the blob axis ratio, angles and centroid distance when the person is not in the lying pose.

Table 4.13 Performance results using state-based algorithm

Dataset 569	
TP	151
TN	360
FP	55
FN	4
Accuracy	89.6%
Sensitivity	97.4%
Specificity	86.7%

There are several false positives created when the subject person enters the room and walks past the mirror just after image 60. Figure 4.18 shows the image sequence and it can be seen that the reflection causes a distortion in the blob as she is walking past. Table 4.14 shows that these false positives are triggered by the variation in the axis ratio along with the centroid distance or the angle variation.

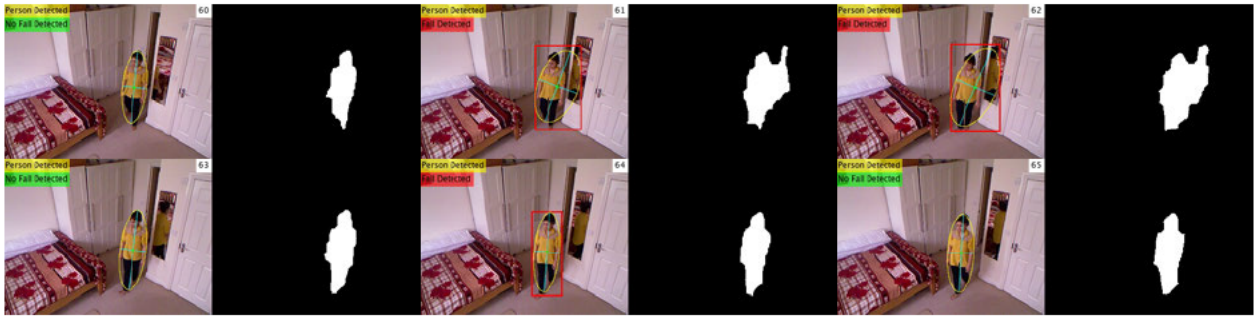

Figure 4.18 Image sequence showing false positive due to mirror reflection

Table 4.14 Image data showing false positive information from reflected image

Image No.	Label	Fall	Centroid Distance	Bounding Box Ratio	Axis std dev	Angle std dev	Orientation angle
60	1	0	2.732659296	2.363956042	0.197504685	0.019028472	1.469677749
61	1	1	10.38606453	5.155535377	0.58608572	0.111508425	1.222855562
62	1	1	2.415900064	5.178207962	0.618893077	0.144759744	1.172225362
63	1	0	19.31768921	10.70655127	0.574528314	0.1355028	1.449189744
64	1	1	4.860616315	8.864735197	0.573296313	0.134281549	1.486908975
65	1	0	3.774489666	9.317598398	0.570729326	0.142644328	1.515034037
66	1	1	4.276079915	4.303728632	0.441190864	0.137286892	1.56111323
67	1	0	8.653419571	5.567996384	0.209784592	0.037532722	1.524483181

Random false positives are also created later in the image dataset that starts at image 297 and continues on and off for the next several frames, see Figure 4.19. This is due to the subject person walking out from behind an occlusion, which is the bed in this case. As her entire body comes into view, the ellipse major axis expands and triggers the standard deviation threshold as well as the centroid distance threshold, thus creating a false positive.

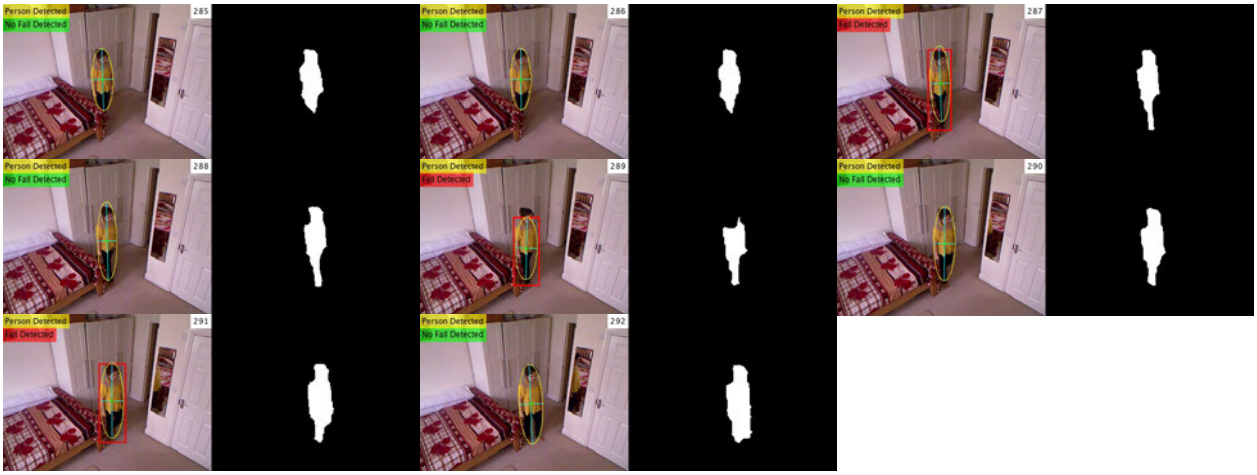

Figure 4.19 Image sequence showing false positives caused by an occlusion

One of the largest causes of false positives in dataset 569 is the series of images just before the subject person lies down onto the bed for the last time after image 407. In this image sequence, shown in Figure 4.20, the person is walking towards the head of the bed in the lower portion of the camera frame. As she reaches the head, she turns and then proceeds to lay down on the bed. This sequence has multiple effects on the algorithm. Firstly, the bottom portion of the person's body is not within the frame, so the long axis of the ellipse is automatically shortened which affects the axis ratio standard deviation. Secondly, as the person turns from a side on image to one where she has her back facing the camera, the ellipse minor axis increases, further exacerbating the axis ratio variation. Lastly, the orientation angle variation increases due to the person changing directions as she moves from standing to lying.

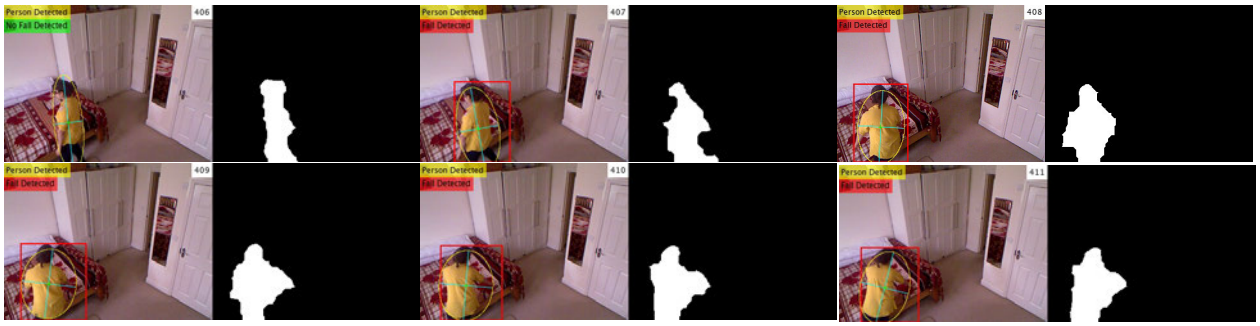

Figure 4.20 Image sequence showing false positives due to part of person being out of frame

The remainder of the false positives are those surrounding the change in pose from standing to lying. They can range from one or two frames to several frames both when the person goes into the lying pose and when they move into a standing pose. As the detection algorithm is tuned to detecting the movement changes that precede and follow the fall event, these types of false positives should be expected, and an acceptable amount will be tolerated.

4.1.7 Revised state-based fall detection

The previous state-based fall detection algorithm was successfully able to detect a change in pose to lying, maintain the fall state while the person remained in the lying pose, then reset the fall state once the person returned to the standing position. There were still many images, however, where the fall detection algorithm has incorrectly detected a fall. In this section, the state-based fall detection algorithm will be modified further so that most of these false detections can be eliminated.

For this algorithm, centroid distance has been removed. This parameter did not improve the performance of the algorithm because the distance changes between each frame is inconsistent. A significant change in distance can be observed when the person is just walking normally around the room as well as during a change in pose. Therefore, it has the potential of contributing to false fall detections as a suitable threshold is impossible to calculate. This is shown by analysing Table 4.10 and Table 4.14. A fall event in Table 4.10 shows that the centroid distance travelled around 3 to 5 pixels. A person walking normally in Table 4.14 covers a similar distance.

When a person falls parallel with the camera vertical axis, it is expected that the variation of the axis ratio should occur mainly in the major axis with minimal changes in the minor axis. For this reason, a fall should only be considered when the axis ratio standard deviation is above the threshold and the minor axis standard deviation is below a threshold. This should help filter the out the non falls where the minor axis is varying significantly, but the major axis is staying relatively the same.

Figure 4.21 Image sequence showing false detection due to minor axis variation

Take image sequence 393 to 397 in Figure 4.21 and Table 4.15 for example. The person is standing side on to the camera and as they begin to move, they turn slightly to be more front on to the camera. This subtle movement causes the minor axis of the ellipse to widen and trigger a false fall detection on axis ratio standard deviation.

Another issue with axis ratio variation occurs when the subject person is in the lower portion of the image. The person becomes partially removed from the image and as a result, the blob is truncated vertically. This means that the major axis of the ellipse will vary significantly whilst the minor axis stays relatively the same. To alleviate this, separate thresholds will be implemented when the person is operating in the lower portion of the frame. Image 400 (Figure 4.22) shows the last frame before parts of the lower body go out of the image. The centroid location at this point is 147, 173. Image 408 (Figure 4.23) shows the person at the bottom of the

frame, this time with the lower portion of their body outside the image. The centroid location for this image is 67, 184. Therefore, fall detection thresholds should be adjusted when centroid y-axis location is below 170 pixels. Image size is 360 x 240.

Table 4.15 Image sequence data showing variation in major and minor axis

Image No.	Label	Fall	Axis std dev	Major axis	Minor axis	Axis ratio
393	1	0	0.096289691	141.603448	30.9939514	4.568745
394	1	0	0.121720427	142.145179	33.6082228	4.229476
395	1	0	0.157821637	141.99376	34.3641865	4.132027
396	1	0	0.387806034	135.541973	39.1275832	3.464103
397	1	1	0.412956157	135.770682	37.8690168	3.585271

Table 4.16 shows the minor and major axis information that should help derive some thresholds for when the person is partially below the bottom of the frame. The table shows that there is a significant change in the length of the major axis when the person is partially out of frame which also impacts the axis ratio standard deviation.

Table 4.16 Image data between image containing entire person (400) and partial person (408)

Image No.	Label	Fall	Axis std dev	Major axis	Minor axis	Axis ratio
400	1	0	0.164425909	147.676259	48.1004694	3.070163
408	1	0	0.880640959	105.383044	84.1834256	1.251827

Figure 4.22 Lowest part of image where entire body of person is shown

Figure 4.23 Image showing partial shape of person in lower part of frame

To help the algorithm understand when a person is clearly standing in an upright pose an upright axis ratio threshold has been developed. This will help eliminate false positives caused by variations in the ellipse major axis from anomalies such as occlusions and partial body capture. This parameter will be included in the algorithm to only consider falls if this ratio is less than the threshold when looking at axis ratio standard deviation type falls.

With this information the revised state-based fall detection algorithm works as follows.

- If the axis ratio standard deviation is above the threshold AND the axis ratio is less than the upright axis ratio threshold AND the ellipse minor axis standard deviation is below the threshold AND the orientation angle is within upright limits, then a fall is detected.
- If the orientation angle standard deviation is above the threshold and the orientation angle is outside upright limits, then a fall is detected.
- Fall detection is removed when the orientation angle is within upright limits AND the upright axis ratio is greater than the threshold.

The thresholds have also been revised for this dataset and have been set at the following levels.

- Orientation angle standard deviation – 0.1 radians (5.7º)
- Axis ratio standard deviation – 0.4
- Upright orientation angle limits – 1.05 to 2.1 radians (60º to 120º)
- Upright axis ratio threshold – 1.7
- Minor axis standard deviation – 5

When the centroid is located in the lower portion of the image the revised thresholds are as follows.

- Upright axis ratio threshold – 1.05
- Upright orientation angle limits – 0.96 to 2.18 radians (55º to 125º)

The performance of this algorithm improved slightly from the previous algorithm, however, it achieved the goal to reduce the amount of false positives being experienced by the algorithm running on dataset 569. The results in Table 4.17 show that the revised algorithm was able to reduce the number of false positives by 35 which has improved the overall accuracy and the specificity. Upon inspection of the data the only remaining false positives are those that are as a result of the movement detected by the fall detector prior to a fall and before they get to a standing position after the fall.

Table 4.17 Results of revised state-based algorithm on dataset 569

Dataset 569	
TP	151
TN	395
FP	20
FN	4
Accuracy	95.8%
Sensitivity	97.4%
Specificity	95.2%

This algorithm was then applied to other datasets that had an appropriate number of training frames at the beginning of the dataset. The results of the algorithm on these datasets in Table 4.18 show that all of the datasets show fairly good sensitivity, however, the accuracy and specificity yield mixed results. This is mainly due to the presence of an excessive number of false positives, which tells us that while the algorithm is good at detecting falls, it is overly sensitive to changes in pose that do not represent a fall.

Table 4.18 Results of revised state-based algorithm on other datasets

	Dataset 758	Dataset 786	Dataset 832	Dataset 1954	Dataset 2123
TP	76	187	286	157	572
TN	665	357	436	667	992
FP	17	242	110	120	432
FN	0	0	0	33	0
Accuracy	97.8%	69.2%	86.8%	84.3%	78.2%
Sensitivity	100.0%	100.0%	100.0%	82.6%	100.0%
Specificity	97.5%	59.6%	79.9%	84.8%	69.5%

Dataset 758 appears to have good overall performance metrics and when the video is analysed it can be seen that the bulk of the false positives come in the frames just prior to the person going into a lying pose and also in the frames just after. There is also a series of false positives generated when the person changes to a sitting pose on the bed. The example in Figure 4.24 shows that the false positives are due to poor foreground extraction from the foreground detector which has, in turn, caused a distortion in the ellipse and generated the false

Figure 4.24 False positive caused by poor image segmentation in dataset 758

detection. In dataset 786 there a couple of scenarios that have led to the high number of false detections. The first, shown in Figure 4.25, is when the person of interest sits down just in front of the couch in a semi-slouched pose. While the dataset label for this image has this as a sitting pose, the fall detection algorithm has interpreted this as a fall due to the characteristics of the pose. The second false positive scenario occurs when the person sits up after lying pose, shown in Figure 4.26. Despite the orientation angle being close to upright, the axis ratio is not above the threshold to be able to reset the fall detection flag. While these two scenarios are not

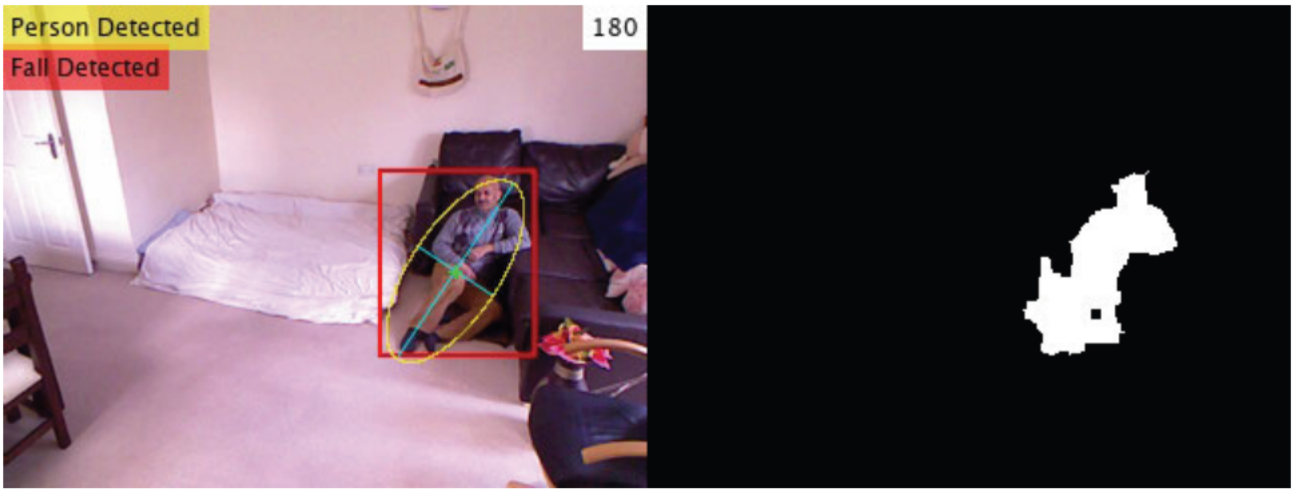

Figure 4.25 False positive caused by incorrectly identified lying pose in dataset 786

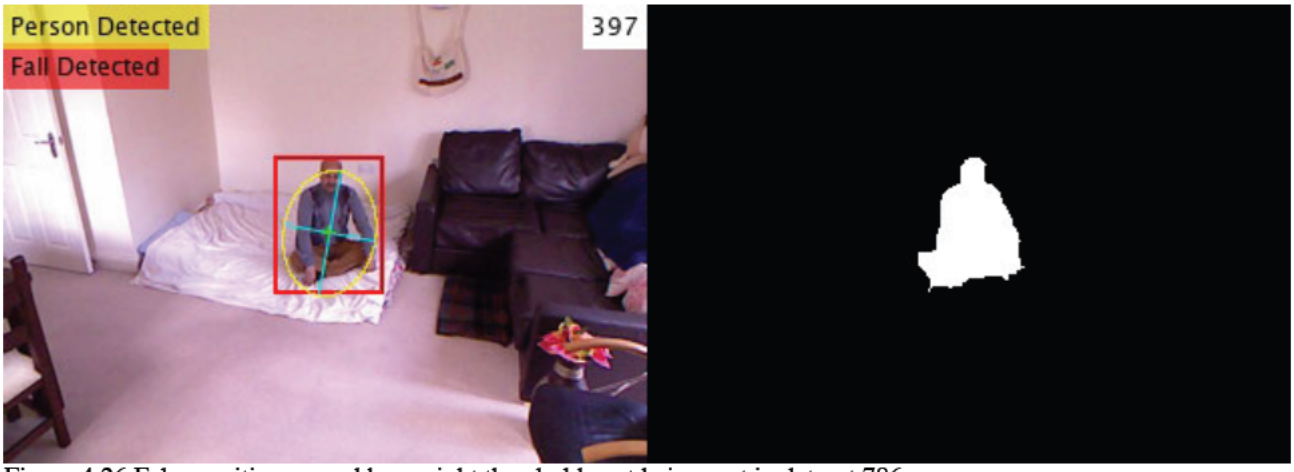

Figure 4.26 False positive caused by upright thresholds not being met in dataset 786

ideal, their presence is not disastrous. In this dataset, the image in Figure 4.25 has been labelled as a sitting pose, however, there is a high likelihood that a fall at this location could end up in this type of pose. In Figure 4.26, the person may have sat up after the fall, but still may not be able to get to their feet and would therefore still need assistance. Dataset 832 suffers from false positives due to both poor foreground extraction as well as an incorrectly identified pose due to the upright thresholds not being met after a lying pose. The image shown in Figure 4.27 shows that the person is wearing a dark pair of pants that have blended in with the dark background caused by the shadowing around the couch. This lack of contrast has made it difficult for the foreground detector to detect the legs of the person and as such, the resultant foreground mask has been truncated. The fall detection algorithm has then picked up on the change in axis ratio and generated an incorrect fall detection. Like dataset 786, dataset 832 has also maintained a fall detection after the person has sat up after a fall. Figure 4.28 shows this pose and like dataset 786, the error should be seen as a positive attribute because the person has not fully recovered from the fall. Dataset 1954 is unique from the other datasets discussed here due to the lack of lighting in the room shown in Figure 4.29. While the foreground detector has done a surprisingly good job in detecting the person in darkened the room, it was unable to locate the person at all when they were located in front of the dark coloured couch. This means there is a high potential for a fall to go unnoticed by the detection algorithm and is a genuine weakness of the system. This dataset also suffered

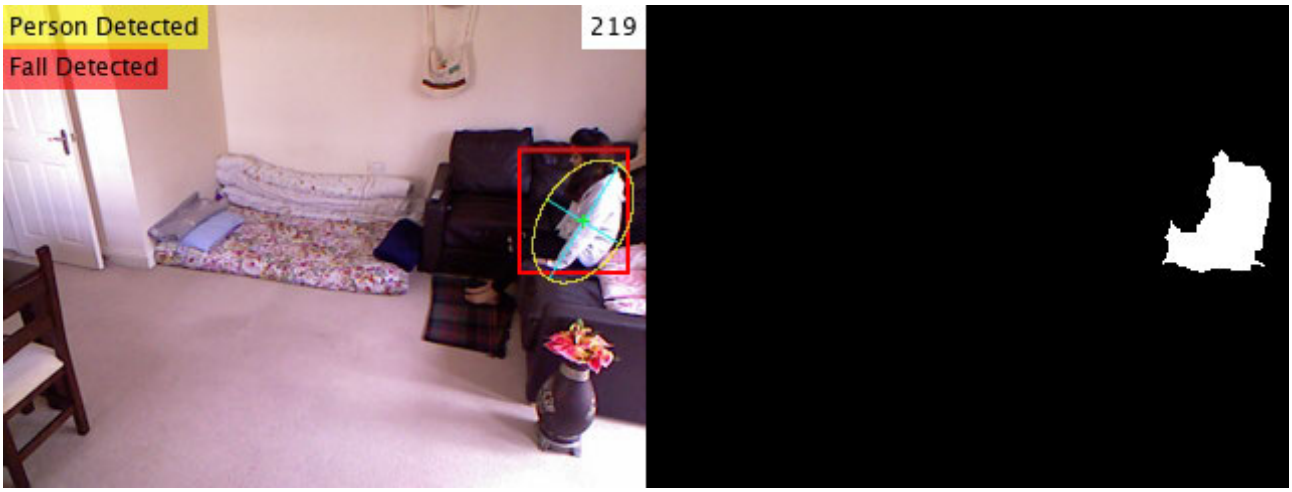

Figure 4.27 False positive due to poor foreground extraction in dataset 832

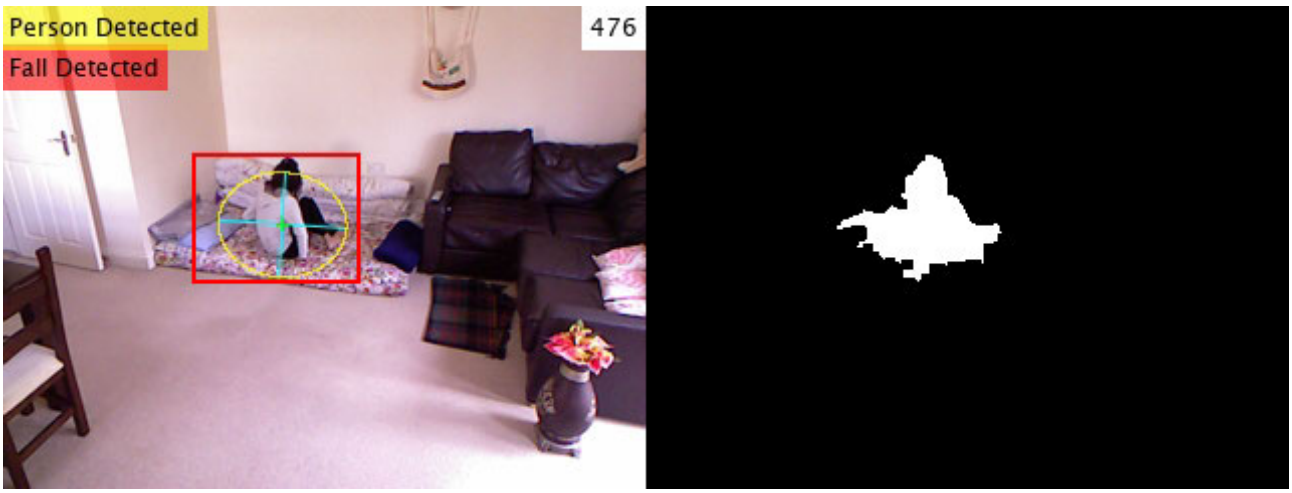

Figure 4.28 False detection due to upright thresholds not being met in dataset 832

Figure 4.29 Person not detected due to dark background in dataset 1954

from a series of false negatives despite good foreground extraction. The image sequence shown in Figure 4.30 shows the person slowly lying down onto the mattress on the ground over a series of twelve frames. Due to the speed of this change of pose, the thresholds for axis ratio and orientation angle have not been triggered and a fall is not detected. Once again, this error can be seen as a positive feature of the algorithm. If applied correctly, only changes in pose that occur at a certain rate may be deemed a fall, not just any transition to a lying pose as has been done in this algorithm. Dataset 2123 is also a unique dataset in that there are variations in lighting intensity throughout the room due to light streaming in through a window. Exacerbating this issue is that the camera automatically adjusts the exposure setting as the person moves around the room causing the foreground detector to create erroneous foreground extractions. This, in turn, generates a multitude of false

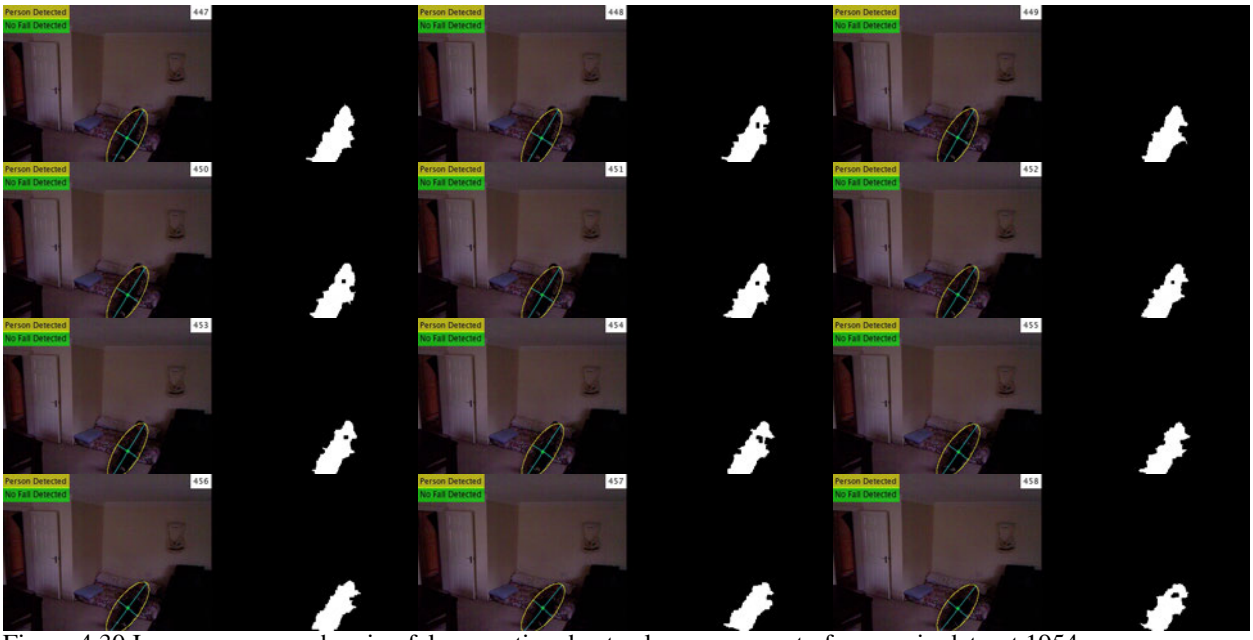
Figure 4.30 Image sequence showing false negative due to slow movement of person in dataset 1954

detections. Take Figure 4.31, for example. In the image on the left the person is squatting prior to sitting down onto the mat and the foreground mask is indicative of that pose being carried out. The image on the right is three frames later where the camera exposure setting has automatically adjusted and the image has become brighter. The result of this change has caused the foreground detector to include the mat and pillow as part of

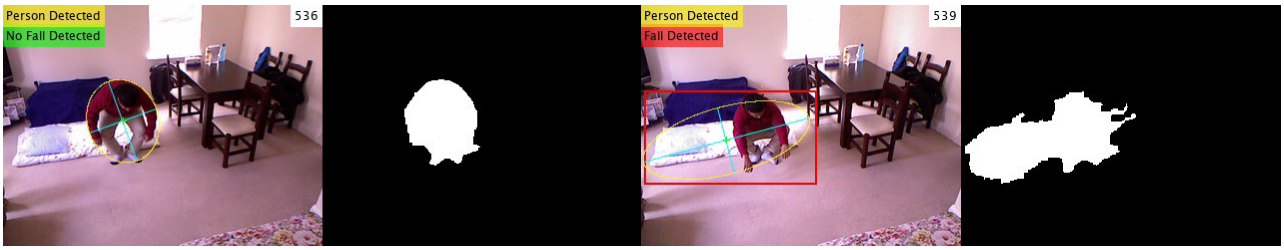

Figure 4.31 Images showing change in camera exposure and effect on foreground mask in dataset 2123

the foreground mask which has distorted the shape of the ellipse and caused the fall detector to trigger a fall. This issue occurs several times throughout this dataset and usually lasts until the exposure returns to the setting where the foreground detector was trained on. This can be in the order of 60 frames. This dataset also had a segmentation issue at one point in the image sequence where the person sits at the dining table. The fall detection algorithm successfully detects a no fall situation until the foreground detector spuriously fails to effectively segment the image and only the legs of the person are successfully extracted into the foreground, see Figure 4.32. The effect that this has on the fall detection algorithm is obvious as the resultant ellipse is distorted from the full body ellipse.

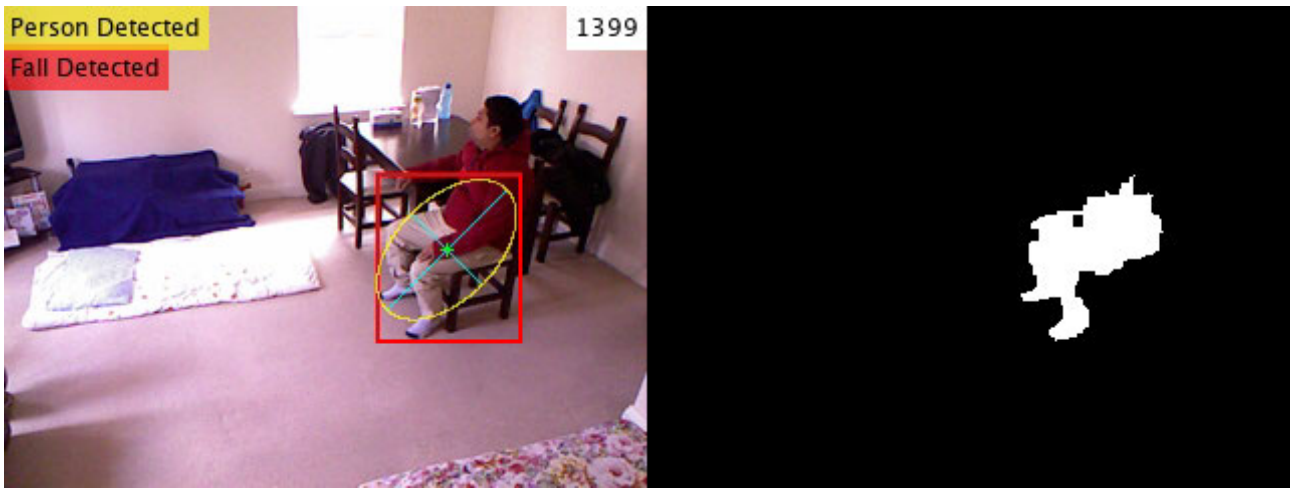

Figure 4.32 Image showing error in foreground extraction in dataset 2123

The performance of the final fall detection algorithm on all the relevant datasets is shown in Table 4.19. Overall, the algorithm has shown the capacity to detect falls effectively and reliably albeit with a few common errors. To be able to effectively analyse the shape changes and discriminate between a fall and a no fall, the fall detection algorithm requires a solid foreground mask. Poor foreground extraction has been seen to be one of the major problems with false detections in all the datasets and must be addressed as a priority. There are several reasons the foreground detector was unable to generate a consistent foreground mask and one of these was lighting variations. Some of the datasets were dark and it was seen that the foreground detector struggled to segment the person in certain locations. On the other hand, when the room had lighting variations, including bright spots, the detector was unable to produce a consistent foreground extraction. One of the reasons for this was automatic exposure adjustments by the camera. If foreground extraction is going to be the method used for fall detection, then a method must be adopted that allows for the lighting changes and exposure settings be manually adjusted as the light changes throughout the day. Lastly, the only series of false negatives were produced when the person changes their pose slowly when laying down onto the mat. These could be

eliminated if the labels for the datasets provided a label that described a rapid change in pose to lying. A real fall would obviously generate a much larger change in parameters than one where the person just lies down. By adjusting the thresholds accordingly, the algorithm would be able to differentiate between the two.

Table 4.19 Overall results of revised state-based algorithm

Overall Results	
TP	1429
TN	3512
FP	941
FN	37
Accuracy	83.5%
Sensitivity	97.5%
Specificity	78.9%

4.2 Ethical Considerations

As with any engineering project, researchers and developers must consider all aspects of the design, particularly the impact that the development will have on society right throughout the design's lifecycle. This Engineering Research Project has been researched and developed with the intention of improving the livelihood of elderly people and enabling them to live independently in their own home into their final years. Unfortunately, there are many aspects to this proposal that could have a negative impact on the lives of the elderly if not addressed properly.

As seen in the literature review, there were several common issues that researchers found concerning the ethics of using computer vision with any project where people were involved whether in the public or private space. Of these issues, the most concerning and the most relevant to this project are the privacy, security, consent and trust elements. In terms of privacy, Senior et al. (2003) list several aspects that should be contemplated with computer vision applications. They believe that what data is present on the system is an aspect that should be considered. If the available data in the captured image is limited, there is less chance that privacy can be breached. Techniques that can help with this include, only limit the capture area to insensitive areas, allow the camera to be easily obstructed or switched off by the user or use a low resolution or unfocussed camera lens. The form of the data should be in a digital form, encrypted and only accessible to those with appropriate authorisation. Online hackers have the ability to gain access to internet cameras and obtain all sorts of personal information about the user, so a solid level of encryption is vital here. This will not only help deal with who can see the captured data but will also help protect the entire system. Senior et al. (2003) believe that the most crucial privacy aspect that should be considered is how raw the data is. They argue that by masking out privacy invasive elements in the video, the privacy can be most effectively enhanced.

Often considered alongside privacy, security is another issue with computer vision systems. Lauronen (2017) found that espionage and identity theft are the two most concerning problems with computer vision systems. Espionage is the act of spying with the intention of stealing information from an individual without their consent while identity theft is stealing the identity of another for malicious purposes and gaining advantages from them. Computer vision can reveal a great deal about the person including their gender, height and ethnicity which can all be used to obtain their identity information more easily. Malicious attacks are also closely aligned to security issues and Lauronen (2017) defines this as where criminals attempt to hack or break into the computer system to spread malicious data including malware and other unwanted material. They cite Rinner and Winkler (2014) who warn that the confidentiality of the image data must be preserved throughout its lifecycle.

Understanding of the intention computer vision systems and how they will operate can be difficult for people to understand, particularly for the elderly. This is where consent can become difficult. Informed consent is something that Coupland et al. (2009) have found to be one the more significant ethical concerns. They have found that informed consent builds a level of trust with the user and it important for protection of privacy. They believe that an assistive technology targeted at the elderly needs to have mechanisms in place that deal with helping the elderly understand how the system works and how their data is going to be used so they can provide a firm level of informed consent.

Consent is built from trust of the system. If the elderly person understands how the computer vision system works and can see the benefits it will provide to their life without subjecting them to any risk, then they will trust the technology (Coupland et al. 2009). Trust of computer vision systems must be built by addressing the issues mentioned above. Skirpan and Yeh (2017) have found that issues with privacy and bias from computer vision systems has led to a decrease in public trust of the technology.

This project has highlighted the potential ethical issues with the proposed system from the beginning and has been addressing them throughout the research phase to the development of the algorithm. Consideration of ethical issues will need to continue through to development of a prototype and testing on some live data. Following on from this, if there was a possibility that the system could be implemented in the public arena, then consideration of ethical issues must be addressed for the entire lifecycle of the product. It was seen that a publicly available dataset was used to test the algorithm on. By using a publicly available dataset any requirement for consent does not need to be considered and any ethical problems can be avoided. This worked well for initial configuration and testing of the algorithm, however, to be able to build and test a working prototype, consent must be gained from the relevant parties.

A working model of this system will need to take care of the privacy and security of the person whose home the system is installed in, particularly in sensitive areas like bathrooms and bedrooms. This can be achieved by removing the ability for anyone to access the video stream created by the camera. Encryption and authentication techniques can be implemented for any access of the system both locally and via remote connection. However, for ultimate privacy and security, the video does not need to be streamed over the internet at all. The requirement for this system to be successful is to alert relevant parties, be that friends,

family, neighbours, or medical personnel, so there is no real requirement for the live video to be accessible remotely.

Finally, for this system to gain the approval of the elderly people who will ultimately benefit from it, they need to not only understand how this system can assist them but also all the risks associated with it. Only through knowledge can trust be established.

Chapter 5 – Conclusions

5.1 Ageing Population

This research project has highlighted that the population of Australia and the world is slowly getting older. It was also discovered that as people approach their senior years, they are choosing to remain in their own homes, a predicament referred to as 'aging in place'. This phenomenon will place further strain on a health care system that is already under stress, particularly in rural areas.

Elderly falls were shown to be one of the major causes of injury in elderly people living at home and the probability of death increased when an elderly person was unable to return to their feet after suffering a fall. A situation also known as the 'long-lie'. The major objective of this project was to develop a fall detection algorithm using computer vision that could not only detect a fall, but also ensure that the person did not spend any longer on the ground than was necessary.

5.2 Literature Review

It was discovered in the literature review that there are already many methods for detection of human movement, including fall detection that have been researched. Of these methods, most of them could be categorised into three main classifications. One classification is wearable technology, where the person wears a device, such as a watch or necklace, and the fall is detected by in-built sensors in the device. Another class were ambient devices or sensor networks installed in the home. These ambient devices monitor the rooms using technologies such as vibration, infra-red and appliance monitoring and will feed the data into some type of logic solver which have in-built algorithms that will decide if a fall has occurred. Finally, computer vision technologies use cameras installed in the home to monitor human movement. The images obtained from the cameras are fed into a computer where they are analysed to decide if a fall has occurred. Some of the techniques for analysing the images use foreground extraction and then analysing the changes in the shape produced from the extraction for determining changes in pose, including fall detection. Other techniques used depth imagery to extract skeletal models of the person. The skeletal model can then be analysed as the person moves around the room and falls detected by certain skeletal poses. Some of the more advanced methods of fall detection use machine learning and deep learning techniques. This was found to be done using Artificial Neural Networks (ANNs) with a Support Vector Machines (SVM) or other machine learning techniques, such as the You Only Look Once (YOLO) system.

5.3 Ethics

As with any engineering innovation, the impact of the technology, both positive and negative, must be considered. It was discovered in the literature review that there are many facets of the ethical impacts to the general community when using computer vision technology. Some of the major concerns were found to be the security and privacy of the data that is captured by computer vision technology. The researchers found that

there are several approaches to ensuring that the system meets the moral and ethical standards of the community. This included setting ethical standards at the developer level where companies or individuals develop their own internal principles and assess the potential impacts of the technology from development through the entire lifecycle of the product. Standards also need to be set at the Government level. It was revealed that there has been very little establishment of standards worldwide when it comes to AI implemented for use in society. A gap that definitely needs to be filled as more and more AI systems are introduced into the future. Finally, the development of these standards need to be driven by the wider community. It is only once all the risks of AI systems are understood and how these risks are mitigated can the community begin to trust AI and take advantage of all the benefits it has to offer.

5.4 Algorithm Development

The development of the fall detection algorithm went through several iterations before arriving at the final algorithm presented in chapter 4. All the iterations have utilised the foreground extraction method to create a foreground mask of the RGB image. The fall detection algorithms then analysed the shape of the foreground mask and different methods were employed to determine if the person in the image had changed to a lying pose. This change to a lying pose was interpreted by the algorithm as a fall and a flag was added to a spreadsheet for comparison with the image label.

The first algorithm iteration used background subtraction from a reference image to produce a foreground mask. Then a bounding box was applied to the mask and the fall detection algorithm analysed the box for changes in aspect ratio to determine if a fall had occurred. The main problems with this revision were that the background subtraction was ineffective at properly segmenting the image and the bounding box was either not able to capture all type of falls or would incorrectly detect a fall.

A more reliable foreground extraction was found using the MATLAB foreground detector function in conjunction with the blob analyser function. The latter function also produced several outputs that were used by the fall detection algorithm for fall detection analysis. An approximated ellipse was one of the outputs created and it was with this that most of the shape analysis was conducted. The major and minor axes of the ellipse were analysed for changes in ratio, the major axis orientation angle of the ellipse was also analysed for changes and the centroid of the ellipse was analysed for distance travelled between frames. To getter a better understanding of the variability of the parameters, the standard deviations of the axis ratio and orientation angle were analysed using the previous five image frames. The algorithm required some threshold levels for each of the parameters so it could tell when a fall was occurring. To achieve this the image sequence was run with arbitrary thresholds in the algorithm and each parameters value in the current image was output to a spreadsheet. By analysing the change in parameter values around the images where the change in pose occurred, threshold values were able to be determined.

It was found that while the foreground detector was generally good at producing reliable foreground extraction, there were several occasions where it produced some inconsistent blobs due to detection of shadows. To

combat this, the RGB image was converted to an HSV image prior to image segmentation being carried out. After this, the fall detection algorithm produced a lot more reliable results, however, there were still quite a few false detections due to the algorithm not maintaining the fall status after the person had transitioned to a lying pose and had not yet stood up again.

At that stage the fall detection algorithm was changed to a state-based algorithm where if any of the fall conditions were met the state would change to the fall state. It would then remain in a fall state until an upright condition was met. This upright condition was achieved by setting appropriate thresholds in the orientation angle as well as axis ratio. These threshold levels were found using a process similar to the fall detection thresholds. After this, the performance of the algorithm was vastly improved, however, there were still several instances of false positives due to a variety of reasons. After some tweaking of the algorithm, most of these false positives were eradicated for the dataset that the algorithm was developed on. The final fall detection algorithm was then run on a few of the other datasets that had enough initial frames for the foreground detector to be trained on. After analysis of the literature, it is believed that this state-based fall detection with upright reset is unique from other fall detection techniques.

5.5 Results

It was discovered that, overall, the fall detection algorithm was effective at detecting the change in pose and at maintaining the detection during a prolonged lying event. So, in answer to the second research question presented in chapter 3, shape analysis appears to be an effective method for fall detection. However, there were many occasions when the fall detection algorithm incorrectly detected a fall. It was discovered that one of the main reasons for these incorrect detections, among others, was from poor foreground extraction and the main cause of this was lighting variations in the images. As a result, it was established that to produce a reliable fall detection using shape analysis, a solid foreground mask must be produced. Therefore, in answer to the first research question posed in chapter 3, foreground extraction could only be considered if changes in lighting conditions are accounted for.

Unfortunately, due to reasons mentioned above, the algorithm did not meet the benchmark setting of 90% stipulated in Chapter 3 and in accordance with other similar systems uncovered in the research conducted by Gutiérrez et al. (2021). Despite this, it is this author's belief that this project has highlighted that computer vision can be a viable system that is worthwhile of further research.

5.6 Future Works

At the beginning of this research project, it was anticipated that a working prototype, using a camera connected to a microprocessor, would be able to be produced. The microprocessor, connected to a network, would have been able to monitor the fall detection output and upon detection of a prolonged fall, raise an alarm. Unfortunately, the time was not available to develop and implement this system so this could be a personal exercise following graduation. Further work also needs to be conducted on the image processing and shape

analysis to try and remove the high number of false positives that plagued most of the datasets. This would involve using different datasets with a focus on trying to find datasets that have stable camera exposure settings so that the foreground extraction is not affected. Finally, the future of computer vision appears to lie with machine learning using deep learning techniques. Further development of the fall detection algorithm would be using the datasets supplied by Adhikari et al. (2017b) to train the algorithm using Artificial Neural Networks for detection of lying and other poses.

References

Adhikari, K., Bouchachia, H. & Nait-Charif, H. 2017a, 'Activity recognition for indoor fall detection using convolutional neural network', *2017 Fifteenth IAPR International Conference on Machine Vision Applications*, pp. 81-84.

Adhikari, K., Bouchachia, H. & Nait-Charif, H. 2017b, *Fall Detection Dataset*, viewed 5 May 2021, <https://falldataset.com/>

Alwan, M., Rajendran, P.J., Kell, S., Mack, D., Dalal, S., Wolfe, M. & Felder, R. 2006, 'A Smart and Passive Floor-Vibration Based Fall Detector for Elderly', *2006 2nd International Conference on Information & Communication Technologies*, 24-28 April 2006, pp. 1003-1007.

Anderson, E.M., Larkins, S., Beaney, S. & Ray, R.A. 2018, 'Should I Stay or Go: Rural Ageing, a Time for Reflection', *Geriatrics (Basel, Switzerland)*, vol. 3, no. 3, pp. 49.

Attar, M., Alsinnari, Y., Alqarni, M., Bukhari, Z., Alzahrani, A., Abukhodair, A., Qadi, A., Alotibi, M. & Jastaniah, N. 2021, 'Common Types of Falls in the Elderly Population, Their Associated Risk Factors and Prevention in a Tertiary Care Center', *Cureus 13(5): e14863. doi:10.7759/cureus.14863*.

Australian Bureau of Statistics 2020, *ERP by LGA (ASGS 2019), Age and Sex, 2001 to 2019*, Commonwealth of Australia, viewed 21 May 2021, <http://stat.data.abs.gov.au/Index.aspx?DataSetCode=ABS_ANNUAL_ERP_LGA2019>

Blank, A.L. 2019, *Computer Vision Machine Learning and Future-Oriented Ethics*, viewed 20 September 2021, <https://digitalcommons.spu.edu/honorsprojects/107>

Boutellaa, E., Kerdjidj, O. & Ghanem, K. 2019, 'Covariance matrix based fall detection from multiple wearable sensors', *Journal of Biomedical Informatics*, vol. 94, pp. 103189.

Cookie Robotics n.d., *How to Draw Ellipse of Covariance Matrix*, viewed 20 July 2021, <https://cookierobotics.com/007/>

Coupland, S., Wakunuma, K. & Stahl, B. 2009, *Identifying ethical issues during the development of a computer vision based AmI system: A case study*, viewed 20 September 2021, <https://www.researchgate.net/publication/228908286_Identifying_ethical_issues_during_the_development_of_a_computer_vision_based_AmI_system_A_case_study/citations>

Daim, T.J. & Lee, R.M.A. 2020, 'IR-UWB Radar Sensor Based Object Distance Determination as a Preliminary Input for Human Motion Detection', *2020 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 20-20 June 2020, pp. 201-205.

De Miguel, K., Brunete, A., Hernando, M. & Gambao, E. 2017, 'Home Camera-Based Fall Detection System for the Elderly', *Sensors (Basel, Switzerland)*, vol. 17, no. 12, pp. 2864.

Degen, T., Jaeckel, H., Rufer, M. & Wyss, S. 2005, 'SPEEDY:a fall detector in a wrist watch', *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pp. 184-187.

El-Bendary, N., Tan, Q., Pivot, F. & Lam, A. 2013, 'Fall detection and prevention for the elderly: A review of trends and challenges', *International Journal on Smart Sensing and Intelligent Systems*, vol. 6, pp. 1230-1266.

Foroughi, H., Aski, B.S. & Pourreza, H. 2008, 'Intelligent video surveillance for monitoring fall detection of elderly in home environments', *2008 11th International Conference on Computer and Information Technology*, pp. 219-224.

Gutiérrez, J., Rodríguez, V. & Martin, S. 2021, 'Comprehensive Review of Vision-Based Fall Detection Systems', *Sensors*, vol. 21, no. 3, pp. 947.

Huang, Y., Miaou, S. & Liao, T.-Y. 2009, 'A Human Fall Detection System Using an Omni-Directional Camera in Practical Environments for Health Care Applications', *MVA*.

Kaewtrakulpong, P. & Bowden, R. 2001, 'An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection', *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems*.

Kale, G. & Patil, V. 2016, 'A Study of Vision based Human Motion Recognition and Analysis', *International Journal of Ambient Computing and Intelligence,* vol. 7, pp. 75-92.

Kwolek, B. & Kepski, M. 2015, 'Improving fall detection by the use of depth sensor and accelerometer', *Neurocomputing,* vol. 168, pp. 637-645.

Kwolek, B. & Kepski, M. 2016, 'Fuzzy inference-based fall detection using kinect and body-worn accelerometer', *Applied Soft Computing,* vol. 40, pp. 305-318.

Lauronen, M. 2017, *Ethical issues in topical computer vision applications*, University of Jyväskylä, Jyväskylä, viewed 16 August 2021, <https://www.semanticscholar.org/paper/Ethical-issues-in-topical-computer-vision-Lauronen/e7427cbf5aa50047dbdd9b4d759dca9736f77959>

Li, Q., Stankovic, J., Hanson, M., Barth, A., Lach, J. & Zhou, G. 2009, 'Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information', *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, 10.1109/BSN.2009.46, pp. 138-143.

Liu, Z., Yang, M., Yuan, Y. & Chan, K.Y. 2020, 'Fall Detection and Personnel Tracking System Using Infrared Array Sensors', *IEEE Sensors Journal,* vol. 20, no. 16, pp. 9558-9566.

Lord, S. & Sherrington, C. 2001, 'Falls in Older People: Risk Factors and Strategies for Prevention', *Inj. Prev.,* vol. 9.

Maldonado-Bascón, S., Iglesias-Iglesias, C., Martín-Martín, P. & Lafuente-Arroyo, S. 2019, 'Fallen People Detection Capabilities Using Assistive Robot', *Electronics,* vol. 8, no. 9.

Mathworks 2016, *How to remove shadow from frames extracted with foreground detector?*, viewed 14 July 2021, <https://au.mathworks.com/matlabcentral/answers/266772-how-to-remove-shadow-from-frames-extracted-with-foreground-detector>

Mathworks 2021a, *Image Processing Onramp*, The Mathworks, Inc., Natick, Massachusetts, United States, viewed 2 April 2021, <https://matlabacademy.mathworks.com/R2020b/portal.html?course=imageprocessing#chapter=4&lesson=3&section=3>

Mathworks 2021b, *Image Processing with MATLAB*, The Mathworks, Inc., Natick, Massachusetts, United States, viewed 13 April 2021, <https://matlabacademy.mathworks.com/R2020b/portal.html?course=mlip#chapter=3&lesson=5&section=3>

Mathworks 2021c, *rgb2hsv*, MathWorks, Inc., Natick, Massachusetts, United States, viewed 23 August 2021, <https://au.mathworks.com/help/matlab/ref/rgb2hsv.html#mw_41a782c4-3a02-4a1b-a8ba-249f500a5d39>

Mubashir, M., Shao, L. & Seed, L. 2013, 'A survey on fall detection: Principles and approaches', *Neurocomputing,* vol. 100, pp. 144-152.

Núñez-Marcos, A., Azkune, G. & Arganda-Carreras, I. 2017, 'Vision-Based Fall Detection with Convolutional Neural Networks', *Wireless Communications and Mobile Computing,* vol. 2017, pp. 9474806.

Pricewaterhousecoopers 2020, *Computer Vision Fundamentals for Business Leaders*, viewed 18 August 2021, <https://www.pwc.com.au/consulting/assets/pwc-computer-vision-fundamentals-for-business-leaders.pdf>

Quinn, M.J. 2004, *Ethics for the Information Age*, viewed 21 September 2021, <https://www.semanticscholar.org/paper/Ethics-for-the-Information-Age-Quinn/0e036130b9692c95bae4cc483c28a1c05d2962ac>

Ramachandran, A., Adarsh, R., Pahwa, P. & Anupama, K.R. 2018, 'Machine Learning-based Fall Detection in Geriatric Healthcare Systems', *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 16-19 Dec. 2018, pp. 1-6.

Rinner, B. & Winkler, T. 2014, 'Privacy-protecting Smart Cameras', *Proceedings of the International Conference on Distributed Smart Cameras*, Venezia Mestre, Italy, Association for Computing Machinery.

Rougier, C., Meunier, J., St-Arnaud, A. & Rousseau, J. 2007, 'Fall Detection from Human Shape and Motion History Using Video Surveillance', *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, 21-23 May 2007, pp. 875-880.

Roy, N., Dubé, R., Després, C., Freitas, A. & Légaré, F. 2018, 'Choosing between staying at home or moving: A systematic review of factors influencing housing decisions among frail older adults', *PloS one,* vol. 13, no. 1, pp. e0189266-e0189266.

Senior, A., Pankanti, S., Hampapur, A., Brown, L., Li, Y. & Ekin, A. 2003, 'Blinkering surveillance: Enabling video privacy through computer vision', *IBM Research Report,* vol. 22886.

Skirpan, M. & Yeh, T. 2017, 'Designing a Moral Compass for the Future of Computer Vision Using Speculative Analysis', *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1368-1377.

Standards Australia 2019, *Developing Standards for Artificial Intelligence: Hearing Australia's Voice*, Standards Australia, viewed 15 September 2021, <https://www.standards.org.au/getmedia/aeaa5d9e-8911-4536-8c36-76733a3950d1/Artificial-Intelligence-Discussion-Paper-(004).pdf.aspx>

Stauffer, C. & Grimson, W. 2007, 'Adaptive background mixture models for real-time tracking', *Proceedings of IEEE Conf. Computer Vision Patt. Recog, vol. 2,* vol. 2.

Suryadevara, N., Gaddam, A., Rayudu, R. & Mukhopadhyay, S.C. 2012a, 'Wireless Sensors Network Based Safe Home to Care Elderly People: Behaviour Detection', *Sensors and Actuators A: Physical,* vol. 186, pp. 277–283.

Suryadevara, N., Quazi, M.T. & Mukhopadhyay, S.C. 2012b, 'Intelligent Sensing Systems for Measuring Wellness Indices of the Daily Activities for the Elderly', *Proceedings - 8th International Conference on Intelligent Environments, IE 2012*, Guanajuato, Mexico, Institute of Electrical and Electronics Engineers (IEEE), pp. 347-350.

Tomkun, J. & Nguyen, B. 2010, 'Design of a Fall Detection and Prevention System for the Elderly'.

Töreyin, B.U., Dedeoğlu, Y. & Çetin, A.E. 2005, 'HMM Based Falling Person Detection Using Both Audio and Video', *Computer Vision in Human-Computer Interaction*, Berlin, Heidelberg, Springer, pp. 211-220.

Tran, T.-H., Le, T.-L., Hoang, V.-N. & Vu, H. 2017, 'Continuous detection of human fall using multimodal features from Kinect sensors in scalable environment', *Computer Methods and Programs in Biomedicine,* vol. 146, pp. 151-165.

Vallabh, P., Malekian, R., Ye, N. & Capeska Bogatinoska, D. 2016, 'Fall Detection Using Machine Learning Algorithms', *The 24th IEEE International Conference on Software, Telecommunications and Computer Networks (IEEE SoftCOM 2016)*, Croatia.

Wang, J., Zhang, Z., Bin, L., Lee, S. & Sherratt, R. 2014, 'An Enhanced Fall Detection System for Elderly Person Monitoring using Consumer Home Networks', *Consumer Electronics, IEEE Transactions on,* vol. 60, pp. 23-29.

Wild, D., Nayak, U. & Isaacs, B. 1981, 'How Dangerous are Falls in Old People at Home', *British Medical Journal (Clinical research ed.)*, vol. 282, pp. 266-268.

Williams, A., Xie, D., Ou, S., Grupen, R., Hanson, A. & Riseman, E. 2010, *Distributed smart cameras for aging in place*, viewed 21 May 2021, <https://www.researchgate.net/publication/228783799_Distributed_smart_cameras_for_aging_in_place>

Appendix A – Project Specification

ENG4111/4112 Research Project

Project Specification

For:	Jabin Smith
Title:	Monitoring Elderly Falls in the Home Using Computer Vision
Major:	Instrumentation, Control and Automation Engineering
Supervisor:	Tobias Low
Enrolment:	ENG4111 – EXT S1, 2021
	ENG4112 – EXT S2, 2021
Project Aim:	To develop a computer vision algorithm to monitor the instances of falls in the elderly within their own homes.
Programme:	**Version 1, 9th March 2021**

1. Review existing fall detection technology and recent research
2. Research potential equipment required to build a camera/microprocessor prototype. Determine costs and lead time of purchasing the equipment as well as any other budgetary requirements.
3. Review existing machine vision technologies and research
4. Review existing motion detection and human movement methods
5. Assess potential ethical and other social issues
6. Design effective machine vision algorithm that can determine the instances of falls in human beings
7. Design and conduct experiments to test the effectiveness of the algorithm
    - **a.** Experiment for fall/non-fall events
    - **b.** Experiment indoor/outdoor locations
    - **c.** Experiment in varying light conditions
    - **d.** Experiment varying camera angles (high, low, mid)
    - **e.** Experiment varying camera distances
    - **f.** Other experiments that become apparent during testing
8. Analyse the results of the experiments and make comments and recommendations about the effectiveness of the algorithm

If time and resources permit:

9. Build camera/microprocessor prototype to accept the machine vision algorithm and test with live data

Appendix B – Project Risk Assessment

University of Southern Queensland

USQ Safety Risk Management System

Print View

Version 2.0

Safety Risk Management Plan

Risk Management Plan ID:	Status:	Current User:	Author:	Supervisor:	Approver:
RMP_2021_5595	*Approval Requested*	[redacted]	[redacted]	[redacted]	[redacted]

Assessment Title:	Development and testing of computer vision monitoring system	Assessment Date:	27/05/2021
Workplace (Division/Faculty /Section):	204070 - School of Mechanical and Electrical Engineering	Review Date:	(5 years maximum)

Approver:	*Supervisor: (for notification of Risk Assessment only)*
Tobias Low	*Tobias Low*

Context

DESCRIPTION:

What is the task/event/purchase/project/procedure?	Development and testing of computer vision monitoring system
Why is it being conducted?	Requirement for Engineering Research Project 2021
Where is it being conducted?	Student Home
Course code (if applicable)	ENG4110
Chemical Name (if applicable)	

WHAT ARE THE NOMINAL CONDITIONS?

Personnel involved	Jabin Smith
Equipment	Laptop Computer, Embedded computer/microprocessor system with camera
Environment	Home & office
Other	
Briefly explain the procedure/process	Desktop development of computer vision algorithm inc. testing. Development of computer/micoprocessor prototype

Assessment Team - who is conducting the assessment?

Assessor(s):	Jabin Smith
Others consulted: (eg elected health and safety representative, other personnel exposed to risks)	

Risk Matrix

Probability	Consequence				
	Insignificant No Injury 0-$5K	Minor First Aid $5K-$50K	Moderate Med Treatment $50K-$100K	Major Serious Injury $100K-$250K	Catastrophic Death More than $250K
Almost Certain 1 in 2	M	H	E	E	E
Likely 1 in 100	M	H	H	E	E
Possible 1 in 1,000	L	M	H	H	H
Unlikely 1 in 10,000	L	L	M	M	M
Rare 1 in 1,000,000	L	L	L	L	L

Recommended Action Guide

Extreme:	E= Extreme Risk – Task ***MUST NOT*** proceed
High:	H = High Risk – Special Procedures Required (Contact USQSafe) Approval by VC only
Medium:	M= Medium Risk - A Risk Management Plan/Safe Work Method Statement is required
Low:	L= Low Risk - Manage by routine procedures.

Risk Register and Analysis

	Step 1	Step 2	Step 2a	Step 2b	Step 3			Step 4				
	Hazards: From step 1 or more if identified	The Risk: What can happen if exposed to the hazard without existing controls in place?	Consequence: What is the harm that can be caused by the hazard without existing controls in place?	Existing Controls: What are the existing controls that are already in place?	Risk Assessment: Consequence x Probability = Risk Level			Additional Controls: Enter additional controls if required to reduce the risk level	Risk assessment with additional controls: Has the consequence or probability changed?			
					Probability	Risk Level	ALARP		Consequence	Probability	Risk Level	ALARP
	Example											
	Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	med	Yes
1	Working wit…	Electrocution	Major	Double insulated equipment, skills and knowledge	Possible	High		Isolate equipment prior to working on it, use RCD circuit	Moderate	Unlikely	Me…	
2	Ergonomics	Repetitive strain injury	Minor	Regular stretching/breaks	Unlikely	Low		Conduct appropriate workstation analysis such as chair type, desk height size, document stands etc. Purchase equipment as required	Insignifica	Rare	Low	
3	Ethics	Invasion of privacy, public embarrassment	Moderate	Research is currently private and not publicly available	Possible	High		Consideration of using binary images in the final design, system to be kept in a closed circuit, i.e. not internet accessible, consider further security encryption	Minor	Unlikely	Low	

Step 5 - Action Plan (for controls not already in place)

	Additional Controls:	Exclude from Action Plan: (repeated control)	Resources:	Persons Responsible:	Proposed Implementation Date:
1	Isolate equipment prior to working on it, use RCD circuit				
2	Conduct appropriate workstation analysis such as chair type, desk height size, document stands etc. Purchase equipment as required.				
3	Consideration of using binary images in the final design, system to be kept in a closed circuit, i.e. not internet accessible, consider further security encryption in design of prototype.				

Supporting Attachments

No file attached

Step 6 – Request Approval

Drafters Name: Jabin Smith Draft Date: 27/05/2021

Drafters Comments: For ERP2021 Progress Report

Assessment Approval: All risks are marked as ALARP 0

Maximum Residual Risk Level: Medium - Cat 4 delegate or above Approval Required 2

Document Status: Approval Requested

Step 6 – Approval

Approvers Name: Tobias Low Approvers Position Title:

Approvers Comments:

I am satisfied that the risks are as low as reasonably practicable and that the resources required will be provided.

Approval Decision: Approve / Reject Date: Document Status: Approval Requested

Appendix C – MATLAB Code

Main Program for Dataset 569

```
%% Fall Detector Version 3.4 Dataset 569
%%The Algorithm for the fall detector uses the Matlab function
%%vision.ForegroundDetector to create a foreground mask and then uses
%%vision.BlobAnalysis to analyse the blob created by the detector. The
%%outputs of the blob analysis yield the centroid, area, bounding box,
%%major and minor axes of the blob's ellipsoid and the orientation angle.
%%These outputs are sent to the Possible Fall Detection function to
%%determine if the variables are showing the characteristics of a falling
%%person or a person in a lying pose. The results are compared the dataset
%%image labels in an excel spreadsheet.

%%
%Algorithm Setup

clear;

%create image datastore
ds = imageDatastore(...
    '/Users/jabinsmith/OneDrive/University/ERP2021/Fall Datasets/569/rgb',...
    'FileExtensions',('.png'));

%read spreadsheet containing class label for each frame
class = readmatrix(...
    '/Users/jabinsmith/OneDrive/University/ERP2021/Fall
Datasets/569/569_labels.csv');

%global variables
global blobArea blobCent blobBox blobL1 blobL2 blobAlpha axisRatio boxRatio
global centDist posFall axisStd alphaStd centMean

posFall = 0;                %set possible fall initial value to 0
position = [0 0];          %vector for position of text
position2 = [0 20];        %vector for position of second line of text
position3 = [320 0];        %vector for position of image number
Z = zeros(1,2000);          %register for ellipse coordinates
numIm = numel(ds.Files);    %register for number of files in datastore
SE = strel('square',5);      %structuring element for binary morphology

%image foreground detector outputs the foreground mask
detector = vision.ForegroundDetector('NumTrainingFrames', 43, ...
       'NumGaussians',5,'LearningRate',0.0005,'MinimumBackgroundRatio'...
       ,0.65,'InitialVariance',(35/255)^2);

%Blob analysis tool outputs blob characteristics
blob = vision.BlobAnalysis(...
       'CentroidOutputPort', true, 'AreaOutputPort', true, ...
       'BoundingBoxOutputPort', true, 'MajorAxisLengthOutputPort',true,...
       'MinorAxisLengthOutputPort',true,'OrientationOutputPort',true,...
       'MinimumBlobAreaSource','Property','MaximumCount',1);

%create video file
video_569_V3_4 = VideoWriter(...
    '/Users/jabinsmith/OneDrive/University/ERP2021/Fall
Datasets/569/video_569_4.mp4',...
    'MPEG-4');
video_569_V3_4.FrameRate = 10;
video_569_V3_4.Quality = 100;
```

```
open(video_569_V3_4);

%create figure to display images
f1 = figure('NumberTitle','off','Name','Dataset 569');
hold on;


%%
%loop to read, process and display each image in the dataset
for k = 1:numIm
    image1 = readimage(ds,k);
    imagehsv = rgb2hsv(image1);
    imageBW = detector(imagehsv);
    imageBW = imfill(imageBW,8,'holes');
    imageBW = bwareaopen(imageBW,1490,8);
    %imageBW = bwpropfilt(imageBW,"Area",1,"largest");
    imageBW = imdilate(imageBW,SE);
    imageBW = imerode(imageBW,SE);
    %imageBW = imopen(imageBW,SE);            %image morphology
    %imageBW = imclose(imageBW,SE);
    [area,cent,bbox,l1,l2,alpha]  = blob(imageBW);

    %variables to place outputs from blob analyser
    if isempty(area)
        blobArea(k,:) = 0;
    else
        blobArea(k,:) = area;
    end

    if isempty(cent)
        blobCent(k,:) = [0 0];
    else
        blobCent(k,:) = cent;
    end

    if isempty(bbox)
        blobBox(k,:) = [0 0 0 0];
    else
        blobBox(k,:) = bbox;
    end

    if isempty(l1)
        blobL1(k,:) = 0;
    else
        blobL1(k,:) = l1;
    end

    if isempty(l2)
        blobL2(k,:) = 0;
    else
        blobL2(k,:) = l2;
    end

    if isempty(alpha)
        blobAlpha(k,:) = pi/2;
    else
        blobAlpha(k,:) = alpha;
    end

    if blobAlpha(k,:) < 0
        blobAlpha(k,:) = pi + blobAlpha(k,:);
    end
```

```
    %Get coordinates for ellipsoid axis lines
    [MAxis,mAxis] = axisCoord(blobCent(k,:),blobAlpha(k,:),blobL1(k,:),...
        blobL2(k,:));

    %Get coordinates for ellipsoid
    Z = ellipse(blobAlpha(k,:),blobL1(k,:)/2,blobL2(k,:)/2,blobCent(k,:));

    %Determine possible fall from blob data
    Fall = posFall_V4(blobCent,blobAlpha,blobL1,blobL2,blobBox,k);

    %%
    %Insert image animations
    if Fall == 1 && ~isempty(cent)
        image1 = insertText(image1,position2,'Fall Detected','FontSize',...
            12,'BoxColor','red');
        image1 = insertShape(image1,'Rectangle',bbox,'color','red',...
            'LineWidth',2);

        class(k,3) = 1;     %add fall flag to spreadsheet column
    else
        image1 = insertText(image1,position2,'No Fall Detected','FontSize'...
            ,12,'BoxColor','green');
        class(k,3) = 0;     %add no fall flag to spreadsheet column
    end
    %Add text box when person is in frame
    if ~isempty(cent)
        image1 = insertText(image1,position,'Person Detected','FontSize',...
            12,'BoxColor','yellow');
    end
    %insert bounding box
    %image1 = insertShape(image1,'Rectangle',bbox,'color','red');
    %insert axis lines
    image1 = insertShape(image1,'Line',[MAxis;mAxis],'color','cyan');
    %insert ellipsoid
    image1 = insertShape(image1,'Line',Z);
    %insert centroid marker
    image1 = insertMarker(image1,cent);
    %insert image number
    image1 = insertText(image1,position3,k,'FontSize',12,'BoxColor',...
        'white','BoxOpacity',1,'Anchorpoint','RightTop');
    %display final image with foreground mask
    image2 = imshowpair(image1,imageBW,'montage');

    writeVideo(video_569_V3_4,image2.CData);

    %add image data to spreadsheet for easier analysis
    class(k,4) = centDist(k,:);
    class(k,5) = centMean(k,:);
    if k>5
        class(k,6:7) = blobCent(k,:) - blobCent(k-5,:);
    end
    class(k,8) = axisStd(k,:);
    class(k,9) = alphaStd(k,:);
    class(k,10) = blobAlpha(k,:);
    class(k,11) = blobL1(k,:);
    class(k,12) = blobL2(k,:);
end
%%
close(video_569_V3_4);      %close video file

%create .xlsx file with test results
writematrix(class,'/Users/jabinsmith/OneDrive/University/ERP2021/Fall
Datasets/569/569_V3_4_test_results.xlsx');
```

Fall Detection Function

```
%% Possible Fall Detection function version 4
%%Possible falls are detected if any of the following criteria are met.
%%Fall condition will also latch until blob angle is within upright person
%%limits.

%%- The standard deviation of the angle of the of the blob in the previous
%%five frames is greater that the threshold AND the standard deviation of
%%the axis ratio from the previous five frames is greater than the
%%threshold.

%%- The centroid distance average from the previous 3 frames is greater
%%than the threshold AND the standard deviation of the angle of the of the
%%blob in the previous five frames is greater that the threshold AND the
%%current blob angle is not within the upright person limits

%%- The centroid distance from the previous frame to the current frame is
%%greater than the threshold AND the standard deviation of
%%the axis ratio from the previous five frames is greater than the
%%threshold.

function posFall = posFall_V4(cent,ang,majLen,minLen,box,k)

global axisRatio boxRatio centDist posFall axisStd alphaStd centMean

%%take variables from main program and put into arrays local to this
%%function.
blobCent = cent;
blobAngle = ang;
blobL1 = majLen;
blobL2 = minLen;
blobBox = box;

%Ratio between major and minor axes
axisRatio(k,:) = blobL1(k,:)/blobL2(k,:);
%Ratio between horizontal and vertical sides of bounding box
boxRatio(k,:) = blobBox(k,3)/blobBox(k,4);

%Condition statement to prevent elevated deviations when no blob is present
%in current frame.
if isnan(axisRatio(k,:))
    axisRatio(k,:) = 3.0;
end

%%Condition statement to prevent array errors when no blob is present in
%current frame.
if isnan(boxRatio(k,:))
    boxRatio(k,:) = 0;
end

%%Condition statements to allow calculation of standard deviations and
%%means when not enough frames have been processed
if k == 1
    alphaStd(k,:) = std(blobAngle(1:k,:),1);
    axisStd(k,:) = std(axisRatio(1:k,:),1);
    L2Std(k,:) = std(blobL2(1:k,:),1);
    centDist(k,:) = 0;
    centMean(k,:) = 0;

elseif k > 1 && k <= 5
    alphaStd(k,:) = std(blobAngle(1:k,:),1);
    axisStd(k,:) = std(axisRatio(1:k,:),1);
```

```
    L2Std(k,:) = std(blobL2(1:k,:),1);
    centDist(k,:) = sqrt((blobCent(k,1) - blobCent(k-1,1))^2 + (blobCent(k,2)...
        - blobCent(k-1,2))^2);
    centMean(k,:) = mean(centDist(1:k,:));

elseif k > 5
    alphaStd(k,:) = std(blobAngle(k-4:k,:),1);
    axisStd(k,:) = std(axisRatio(k-4:k,:),1);
    L2Std(k,:) = std(blobL2(k-4:k,:),1);
    centDist(k,:) = sqrt((blobCent(k,1) - blobCent(k-1,1))^2 + (blobCent(k,2)...
        - blobCent(k-1,2))^2);
    centMean(k,:) = mean(centDist(k-2:k,:));
else
    alphaStd(k,:) = 0;
    axisStd(k,:) = 0;
    L2Std(k,:) = 0;
    centDist(k,:) = 0;
    centMean(k,:) = 0;
end

%%Thresholds for fall condition statements
centThresh = 4.0;    %Centroid mean distance threshold
alphaThresh = 0.1;   %Angle standard deviation threshold
axisThresh = 0.4;   %Axis ratio standard deviation threshold
upLeft = 2.09;       %Upright angle threshold for left side of blob
upRight = 1.05;       %Upright angle threshold for right side of blob
ratioThresh = 1.7;
L2Thresh = 5;
%Thresholds for when person is in lower portion of frame
if blobCent(k,2) > 170
    ratioThresh = 1.05;
    upLeft = 2.18;
    upRight = 0.96;
end

%% Fall Condition Statements
% When a fall is detected the algorithm will latch a fall condition until
% it deems that the person has returned to the upright position.

switch posFall

    case 0

        if blobAngle(k,:) > upRight && blobAngle(k,:) < upLeft
            if axisStd(k,:) > axisThresh && axisRatio(k,:) < ratioThresh...
                    && L2Std(k,:) < L2Thresh
                posFall = 1;
            else
                posFall = 0;

            end

        elseif alphaStd(k,:) > alphaThresh ...
                && (blobAngle(k,:) < upRight || blobAngle(k,:) > upLeft)
            posFall = 1;

        else
            posFall = 0;

        end

    case 1
```

```
        if blobAngle(k,:) > upRight && blobAngle(k,:) < upLeft...
                && axisRatio(k,:) > ratioThresh
            posFall = 0;
        end

end

end
```

Axis Coordinates Function

```
function [MAxis, mAxis] = axisCoord(cent, alpha, l1, l2)

    beta = pi + alpha;
    theta = pi/2 - alpha;
    phi = pi + theta;

    X1 = cent(1) + (l1/2) .* cos(alpha);
    Y1 = cent(2) - (l1/2) .* sin(alpha);
    X2 = cent(1) + (l1/2) .* cos(beta);
    Y2 = cent(2) - (l1/2) .* sin(beta);

    x1 = cent(1) + (l2/2) .* cos(theta);
    y1 = cent(2) + (l2/2) .* sin(theta);
    x2 = cent(1) + (l2/2) .* cos(phi);
    y2 = cent(2) + (l2/2) .* sin(phi);

    MAxis = [X1 Y1 X2 Y2];
    mAxis = [x1 y1 x2 y2];
end
```

Axis Ratio Function

```
function axisStd = axisRatio(l1, l2)

axisRat = l1./l2;
axisStd = std(axisRat);

end
```

Centroid Distance Function

```
function centDist = distance(cent)

centDist = sqrt((cent(1,1) - cent(2,1))^2 + (cent(1,2) - cent(2,2))^2);

end
```

Ellipse Draw Function

```
function Z = ellipse(phi,a,b,cent)

Z = zeros(1,2000);
t = linspace(0,2*pi,1000);
Rot = [cos(phi) sin(phi);-sin(phi) cos(phi)];
Coords = [a*cos(t);b*sin(t)];

Y = Rot * Coords;
Y(1,:) = Y(1,:)+cent(1);
```

```
Y(2,:) = Y(2,:)+cent(2);
Z(:,1:2:end) = Y(1,:,end);
Z(:,2:2:end) = Y(2,:,end);
end
```

Appendix D – Fall Analysis Data – Dataset 569

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
1	6	0	0	0	0	1.570796327	0	0		TN
2	6	0	0	0	0	1.570796327	0	0		TN
3	6	0	0	0	0	1.570796327	0	0		TN
4	6	0	0	0	0	1.570796327	0	0		TN
5	6	0	0	0	0	1.570796327	0	0		TN
6	6	0	0	0	0	1.570796327	0	0		TN
7	6	0	0	0	0	1.570796327	0	0		TN
8	6	0	0	0	0	1.570796327	0	0		TN
9	6	0	0	0	0	1.570796327	0	0		TN
10	6	0	0	0	0	1.570796327	0	0		TN
11	6	0	0	0	0	1.570796327	0	0		TN
12	6	0	0	0	0	1.570796327	0	0		TN
13	6	0	0	0	0	1.570796327	0	0		TN
14	6	0	0	0	0	1.570796327	0	0		TN
15	6	0	0	0	0	1.570796327	0	0		TN
16	6	0	0	0	0	1.570796327	0	0		TN
17	6	0	0	0	0	1.570796327	0	0		TN
18	6	0	0	0	0	1.570796327	0	0		TN
19	6	0	0	0	0	1.570796327	0	0		TN
20	6	0	0	0	0	1.570796327	0	0		TN
21	6	0	0	0	0	1.570796327	0	0		TN
22	6	0	0	0	0	1.570796327	0	0		TN
23	6	0	0	0	0	1.570796327	0	0		TN
24	6	0	0	0	0	1.570796327	0	0		TN
25	6	0	0	0	0	1.570796327	0	0		TN
26	6	0	0	0	0	1.570796327	0	0		TN
27	6	0	0	0	0	1.570796327	0	0		TN
28	6	0	0	0	0	1.570796327	0	0		TN
29	6	0	0	0	0	1.570796327	0	0		TN
30	6	0	0	0	0	1.570796327	0	0		TN
31	6	0	0	0	0	1.570796327	0	0		TN
32	6	0	0	0	0	1.570796327	0	0		TN
33	6	0	0	0	0	1.570796327	0	0		TN
34	6	0	0	0	0	1.570796327	0	0		TN
35	6	0	0	0	0	1.570796327	0	0		TN
36	6	0	0	0	0	1.570796327	0	0		TN
37	6	0	0	0	0	1.570796327	0	0		TN
38	6	0	0	0	0	1.570796327	0	0		TN
39	6	0	0	0	0	1.570796327	0	0		TN
40	6	0	0	0	0	1.570796327	0	0		TN
41	6	0	0	0	0	1.570796327	0	0		TN
42	6	0	0	0	0	1.570796327	0	0		TN
43	6	0	0	0	0	1.570796327	0	0		TN
44	1	0	0	0	0	1.570796327	0	0		TN
45	1	0	0	0	0	1.570796327	0	0		TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
46	1	0	0	0	0	1.570796327	0	0		TN
47	1	0	0	0	0	1.570796327	0	0		TN
48	1	0	0	0	0	1.570796327	0	0		TN
49	1	0	0	0	0	1.570796327	0	0		TN
50	1	0	0	0	0	1.570796327	0	0		TN
51	1	0	242.1331651	0.335331506	0.049280994	1.447593843	102.026394	26.5809419	3.838329	TN
52	1	0	3.207382333	0.354840377	0.073221763	1.401147795	107.699306	30.1893388	3.567462	TN
53	1	0	2.679120579	0.326737142	0.076030044	1.408254192	105.85248	32.1758948	3.289807	TN
54	1	0	4.503459742	0.284127278	0.062490165	1.421744847	105.179894	31.935834	3.293476	TN
55	1	0	2.091648325	0.265874215	0.026389015	1.472416361	101.258435	33.0079547	3.067698	TN
56	1	0	5.688398038	0.229416143	0.034603786	1.485911911	116.283759	31.2646594	3.719336	TN
57	1	0	2.189612941	0.211305146	0.036887217	1.503030151	111.340909	33.5728093	3.316401	TN
58	1	0	2.442391588	0.209980543	0.030175947	1.503881104	111.70751	33.4187208	3.342663	TN
59	1	0	1.734153308	0.217028878	0.015247797	1.515829592	110.455123	34.4266523	3.208419	TN
60	1	0	7.00968935	0.670041965	0.045707805	1.390407654	101.07595	56.8444878	1.778113	TN
61	1	0	3.922150164	0.736874353	0.109378291	1.229745013	110.227979	61.3177682	1.797652	TN
62	1	0	2.305358901	0.699851234	0.145727126	1.151865335	126.144853	63.0893323	1.999464	TN
63	1	0	2.803369459	0.533724929	0.159883188	1.076740392	138.445177	68.8472389	2.010904	TN
64	1	0	23.65301032	0.501305901	0.14645175	1.470029674	121.521189	38.8756623	3.125894	TN
65	1	0	4.268896653	0.531213383	0.169812024	1.498999421	118.263609	41.2612292	2.866216	TN
66	1	0	3.95632904	0.503026071	0.193607782	1.542106235	123.712041	40.4399462	3.059154	TN
67	1	0	8.143258394	0.507621684	0.175293969	1.53057577	138.011495	38.872781	3.550338	TN
68	1	0	7.623071951	0.280777322	0.025188636	1.51099409	136.885784	38.2847665	3.575463	TN
69	1	0	4.061477994	0.329483534	0.016396025	1.537374962	137.236518	36.9409471	3.715024	TN
70	1	0	5.816443175	0.230522805	0.010982094	1.523121248	132.040843	39.7798799	3.319287	TN
71	1	0	11.81172155	0.18809288	0.011898999	1.505358583	140.930295	44.1332733	3.193289	TN
72	1	0	6.435770789	0.277671092	0.012627075	1.503696562	136.931335	46.7263044	2.930498	TN
73	1	0	7.433910167	0.306718957	0.027781152	1.579581598	135.264427	47.3598299	2.8561	TN
74	1	0	8.778094199	0.185833117	0.055557609	1.648591372	137.996452	47.9073052	2.880489	TN
75	1	0	9.652915228	0.125562022	0.067032414	1.659812076	138.852114	45.4450848	3.055382	TN
76	1	0	9.01705609	0.068749789	0.056089899	1.585785027	128.557774	43.8534947	2.931529	TN
77	1	0	6.9051915	0.092338332	0.032573179	1.630060578	130.388782	42.2656474	3.084982	TN
78	1	0	6.997476756	0.075977313	0.02993684	1.67131489	127.803094	42.9798789	2.973556	TN
79	1	0	11.83866589	0.132803532	0.030249707	1.652807714	121.403929	44.8111021	2.709238	TN
80	1	0	7.949592034	0.180752719	0.028981942	1.622879679	119.931429	46.2762441	2.591641	TN
81	1	0	8.856605398	0.236927274	0.031936216	1.576850606	113.824845	46.5333194	2.446093	TN
82	1	0	5.693907685	0.206620099	0.041338773	1.565336313	110.807	46.2179336	2.397489	TN
83	1	0	5.238291955	0.126768443	0.032760637	1.582203992	113.48446	47.7206773	2.378098	TN
84	1	0	5.679236943	0.106689476	0.023227577	1.618736611	112.249907	49.6002577	2.263091	TN
85	1	0	6.47235951	0.094612247	0.031059516	1.649292345	111.241918	50.8318599	2.188429	TN
86	1	0	4.793918887	0.158597351	0.047756647	1.698529927	109.925336	56.1014714	1.959402	TN
87	1	0	4.555140835	0.225983661	0.05128385	1.722901891	107.296967	61.3919023	1.747738	TN
88	1	0	4.052093903	0.242221585	0.053293512	1.769538107	107.734144	65.8076753	1.637106	TN
89	1	0	3.174540179	0.237046316	0.048719089	1.783716249	106.59272	69.9322933	1.524227	TN
90	1	0	2.353067848	0.166025018	0.035870844	1.789657762	104.175673	69.0256618	1.509231	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
91	1	0	1.982858161	0.087760054	0.034177357	1.828680516	108.458276	65.8703049	1.646543	TN
92	1	0	0.699200179	0.065184333	0.054835166	1.920929407	112.102677	67.4613062	1.661733	TN
93	1	0	3.432413123	0.063214291	0.106788737	2.06786889	115.56408	74.4406714	1.552432	TN
94	1	0	1.457630235	0.057891033	0.109634856	2.035162639	116.997758	74.6450437	1.567388	TN
95	1	0	0.303784934	0.110947511	0.084895029	1.958243153	121.477117	65.2032562	1.863053	TN
96	1	0	5.865454083	0.11688182	0.05269001	1.983203509	121.183733	69.0808965	1.754229	TN
97	1	0	8.995478213	0.116542173	0.162869093	1.615427226	111.8215	66.2515022	1.687833	TN
98	1	0	16.29833269	0.102246202	0.148093703	1.909675677	122.119334	74.9560982	1.629211	TN
99	1	0	11.64827646	0.077976718	0.202331552	1.482983627	110.118702	64.1437335	1.716749	TN
100	1	0	12.69463228	0.077843107	0.183994628	1.746544055	121.302178	65.1230691	1.862661	TN
101	1	0	4.851520338	0.126849169	0.141328332	1.686370608	122.327135	61.896813	1.976307	TN
102	1	0	3.265191004	0.187516729	0.136564451	1.705493401	126.133686	58.4466063	2.158101	TN
103	1	0	2.616536503	0.183898213	0.106124579	1.791403841	131.32289	59.3330659	2.213317	TN
104	1	0	2.825890953	0.134071136	0.064987776	1.867512633	135.611002	62.4950032	2.16995	TN
105	1	0	3.401838855	0.094479579	0.078075078	1.872069588	134.440374	66.9335485	2.008565	TN
106	1	0	3.806259159	0.069195304	0.063535926	1.85601583	135.210937	62.9446089	2.148094	TN
107	1	0	3.984067265	0.076471147	0.043337711	1.766364184	130.187855	63.4947148	2.050373	TN
108	1	0	4.716042143	0.11288644	0.058911117	1.729307054	132.1323	56.622449	2.333567	TN
109	1	0	0.691861415	0.158418064	0.060524381	1.735259968	131.42573	54.4165374	2.41518	TN
110	1	0	2.914019049	0.131437535	0.045352451	1.766545233	128.341692	55.9587547	2.293505	TN
111	4	0	4.316031482	0.217793367	0.034144673	1.673179479	125.792328	46.1629553	2.724963	TN
112	4	0	3.013232151	0.184159765	0.038641996	1.665498254	121.622806	44.9795923	2.703955	TN
113	4	0	2.029574287	0.169119054	0.043397964	1.65714209	115.551813	47.1640936	2.449995	TN
114	4	0	1.966780047	0.195536185	0.039840496	1.678568183	111.556697	49.2222333	2.266388	TN
115	4	0	0.584317686	0.19422715	0.018170748	1.710274929	110.259309	47.93544	2.300163	TN
116	2	0	1.46608759	0.156661532	0.03706252	1.758763374	110.832023	46.9089587	2.362705	TN
117	2	0	1.520876093	0.089882133	0.046922709	1.781495215	109.426721	50.1196381	2.18331	TN
118	2	0	1.668393795	0.145542004	0.065398059	1.8687115	117.779577	45.0924906	2.611955	TN
119	2	0	3.660170662	0.149078807	0.071819933	1.905340685	120.589368	48.3984361	2.491596	TN
120	2	0	1.857600401	0.21619413	0.076362627	1.963495147	126.938561	45.0200968	2.819598	TN
121	2	0	1.463096258	0.247593468	0.079501009	2.01279291	132.092441	46.0259498	2.869956	TN
122	2	0	3.315843349	0.227722646	0.085709455	2.113012159	143.579183	45.5470895	3.152324	TN
123	2	0	4.571540749	0.211922258	0.086240794	2.130787797	131.191406	45.1100368	2.908253	TN
124	3	0	3.31205671	0.149422258	0.093090233	2.22846213	122.815774	45.532759	2.697306	FN
125	3	1	5.803085337	0.355294859	0.115853525	2.354967578	103.2516	49.2525349	2.096371	TP
126	3	1	4.480965843	0.550256933	0.118882277	2.411953967	88.9863486	53.8806951	1.651544	TP
127	3	1	2.397167562	0.592563717	0.121605921	2.46181038	81.4979384	60.0939863	1.356175	TP
128	3	1	1.903114629	0.531724703	0.1405227	2.071862728	81.8466024	65.4618749	1.250294	TP
129	3	1	2.137580725	0.334641795	0.164166383	2.091968238	80.5282516	67.8983873	1.186011	TP
130	3	1	0.470678976	0.167771903	0.195721667	2.542437155	79.6113008	64.6466998	1.231483	TP
131	3	1	4.788697226	0.133103435	0.285451614	2.825153827	84.5900251	54.290528	1.558099	TP
132	3	1	1.793418569	0.329899453	0.318759567	2.746194672	92.3624757	44.7764398	2.062747	TP
133	3	1	9.315060096	0.441697906	0.255419754	2.499570125	106.640335	46.6026749	2.288288	TP
134	3	1	4.372896925	0.383117684	0.151597633	2.427444569	93.5189307	46.2079551	2.023871	TP
135	3	1	0.390531752	0.237834825	0.168053628	2.425529644	94.0360881	46.5622008	2.01958	TP

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
136	3	1	0.824995371	0.101629487	0.119136904	2.474894799	96.9037879	47.4508223	2.042194	TP
137	3	1	0.291643242	0.10490146	0.03060159	2.486143293	96.8296543	47.8977038	2.021593	TP
138	3	1	1.183588034	0.009763985	0.024645711	2.461604548	98.1646414	48.7645955	2.013031	TP
139	3	1	1.322820522	0.013501032	0.020812438	2.451652112	96.8329591	47.295313	2.047411	TP
140	3	1	0.417073723	0.013138289	0.013652244	2.486193377	96.9324857	47.5274979	2.039503	TP
141	3	1	0.503723369	0.014291663	0.013960341	2.479485127	97.1968538	47.4420863	2.048747	TP
142	3	1	0.572205626	0.024948025	0.012358449	2.46733115	96.8189822	46.3112069	2.090617	TP
143	3	1	2.64448977	0.049077056	0.033417743	2.392957386	102.321804	47.1345292	2.170846	TP
144	3	1	3.304269177	0.049164923	0.033963668	2.440575434	96.226303	47.0143601	2.046743	TP
145	3	1	2.923733427	0.06458279	0.029784564	2.450051766	104.163092	47.236381	2.205145	TP
146	3	1	3.275257255	0.06844998	0.026566235	2.462429304	98.6181103	48.5656699	2.030614	TP
147	3	1	1.3382153	0.088364726	0.033526831	2.376799051	106.202185	47.0988142	2.25488	TP
148	3	1	1.7471003	0.089010273	0.040859118	2.361968263	105.34675	50.4270652	2.089091	TP
149	3	1	1.768756526	0.129880126	0.062828845	2.290303812	105.89772	56.0632347	1.888898	TP
150	3	1	0.975594634	0.119983124	0.083802876	2.214199807	104.176825	52.0081461	2.003087	TP
151	2	0	0.853880632	0.120636265	0.093047893	2.12859604	103.776633	49.3876531	2.101267	TN
152	2	0	0.811447925	0.113544562	0.096331821	2.106362121	107.371337	48.1147112	2.23157	TN
153	2	0	1.151925148	0.117220446	0.072424669	2.106444372	104.19139	48.7764588	2.1361	TN
154	2	0	1.917644852	0.082716208	0.053377194	2.049756667	99.382614	49.1704155	2.021187	TN
155	2	0	9.140086403	0.206075677	0.031030273	2.05581069	124.18412	47.5932548	2.60928	TN
156	2	0	5.281698425	0.222397735	0.029926425	2.035088303	127.403406	50.8425792	2.505841	TN
157	2	0	0.855740625	0.239506075	0.038036829	1.988065646	129.785623	50.7650763	2.556593	TN
158	2	0	0.72962969	0.215571867	0.061803196	1.889543146	121.482836	47.769875	2.543085	TN
159	2	0	1.263596229	0.118772119	0.080994166	1.849127614	114.346668	50.4044588	2.268582	TN
160	2	0	1.49680148	0.136695175	0.090519164	1.787103433	111.193949	49.4534036	2.248459	TN
161	2	0	2.365736591	0.131628967	0.080814761	1.759545687	114.182945	48.3560007	2.361298	TN
162	4	0	3.906114782	0.150208093	0.054772772	1.745618489	122.162379	46.5211506	2.625954	TN
163	1	0	4.554021631	0.134563981	0.037381172	1.756373683	126.902277	53.5662053	2.369073	TN
164	1	0	3.371404795	0.166095474	0.034284504	1.840749631	129.688867	61.0588004	2.124	TN
165	1	0	0.877517567	0.177672793	0.039126251	1.824094586	134.280183	61.8541336	2.170917	TN
166	1	0	1.338735755	0.182688774	0.037051539	1.784875981	133.78754	60.5556405	2.209332	TN
167	1	0	3.163647964	0.106160544	0.044746149	1.717504869	133.203869	55.8352089	2.385661	TN
168	1	0	4.678942746	0.236316266	0.05978145	1.686554225	139.484274	50.3502695	2.770279	TN
169	1	0	2.636147038	0.376831584	0.049582458	1.727111148	147.744058	46.7163966	3.162574	TN
170	1	0	1.356146673	0.450944473	0.034273599	1.760908332	153.001638	44.9480383	3.403967	TN
171	1	0	3.247997805	0.352245579	0.02374757	1.726040998	148.588917	48.1169285	3.08808	TN
172	1	0	4.055188866	0.221007254	0.035993183	1.657103932	142.828039	42.9222107	3.327602	TN
173	1	0	2.768214806	0.160284856	0.043763868	1.64815429	140.960643	47.611694	2.960631	TN
174	1	0	0.847606152	0.222736074	0.049358167	1.762086204	143.88344	51.2578194	2.807053	TN
175	1	0	3.476804465	0.253518342	0.066050859	1.824657664	142.036782	55.1186489	2.576928	TN
176	1	0	3.847551529	0.244273369	0.081672679	1.843135595	141.571442	48.3156464	2.930137	TN
177	1	0	4.193851248	0.136053416	0.072308028	1.829695027	135.774626	47.4648097	2.860532	TN
178	1	0	2.071455789	0.122453373	0.029662833	1.839862799	132.880799	48.9104672	2.716817	TN
179	1	0	6.023556942	0.152578703	0.009917134	1.816015636	123.989109	48.8007679	2.54072	TN
180	1	0	0.856457731	0.139668493	0.029573016	1.90223353	131.487058	49.4561303	2.65866	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
181	1	0	1.588762682	0.104836048	0.045676155	1.934060801	130.397372	49.3044885	2.644736	TN
182	5	0	3.344008031	0.05888642	0.081012871	2.045840476	122.666428	47.1551341	2.601338	TN
183	5	0	3.881806261	0.074470727	0.07871752	1.994382728	132.859956	48.0221921	2.766637	TN
184	5	0	0.878576006	0.067629128	0.052339739	2.011996293	133.356734	48.1734617	2.768261	TN
185	5	0	2.019505546	0.08160617	0.040103015	2.039341625	134.030364	47.6105735	2.815139	TN
186	5	0	1.412186472	0.095390567	0.033215777	2.091699086	135.943031	46.9985041	2.892497	TN
187	3	0	2.249929027	0.050606199	0.053292436	2.139173309	133.803159	48.5424954	2.756413	FN
188	3	0	1.657258723	0.159729105	0.078848972	2.234590912	119.900411	49.401383	2.427066	FN
189	3	1	9.768416393	0.609852103	0.142487975	2.44412915	103.665671	82.888193	1.250669	TP
190	3	1	4.563190452	0.614010245	0.173682965	2.538860274	110.566066	60.8730286	1.816339	TP
191	3	1	3.401854947	0.536115067	0.143532331	2.37758699	106.663961	62.6039509	1.70379	TP
192	3	1	2.136089978	0.375961118	0.104428274	2.316329007	109.78345	59.2679889	1.852323	TP
193	3	1	1.393655515	0.218641894	0.081561593	2.332570841	99.8797547	64.2486606	1.554581	TP
194	3	1	4.75997639	0.187946161	0.080601376	2.433925932	113.577426	53.4865283	2.123477	TP
195	3	1	2.859453835	0.190031079	0.043909804	2.324626897	104.163293	55.4309191	1.879155	TP
196	3	1	2.172757379	0.200152769	0.043123435	2.336567289	102.811096	62.8377744	1.636135	TP
197	3	1	1.931042211	0.217094003	0.078854534	2.18695835	100.698925	63.6800381	1.581326	TP
198	3	1	0.841243954	0.210089946	0.082951797	2.255031057	101.627941	63.7081465	1.595211	TP
199	3	1	0.676187334	0.109783681	0.053968809	2.290036893	102.320131	63.0201912	1.623609	TP
200	3	1	0.437947216	0.022086486	0.048956548	2.276063187	100.381111	63.4019668	1.583249	TP
201	3	1	3.356454758	0.154341095	0.037061897	2.224579767	105.009786	53.0392775	1.979849	TP
202	3	1	1.769502301	0.149535384	0.022715219	2.275053771	103.554069	58.9058124	1.75796	TP
203	3	1	1.977806083	0.147963055	0.022636812	2.256051259	101.694217	54.4885848	1.86634	TP
204	3	1	1.67853823	0.132444216	0.01965734	2.242354487	104.765185	56.632713	1.849906	TP
205	3	1	1.70954225	0.156113526	0.018862157	2.272173522	97.9359354	64.6303226	1.515325	TP
206	3	1	0.789681331	0.13009713	0.036871518	2.348756054	100.190087	60.3275974	1.660767	TP
207	3	1	2.283427692	0.129727637	0.059707178	2.397336333	107.513723	63.3081969	1.698259	TP
208	3	1	3.999502246	0.116966065	0.074206594	2.44016994	114.082408	63.3050921	1.802105	TP
209	3	1	2.490461085	0.106397741	0.082019304	2.214360571	110.391765	61.2258782	1.803025	TP
210	3	1	0.974489815	0.05854412	0.088247361	2.237421902	111.699918	62.709503	1.781228	TP
211	3	1	0.785415061	0.069526035	0.089514498	2.276424988	113.040312	58.9985733	1.915984	TP
212	3	1	5.449308991	0.047354167	0.091284582	2.406444955	116.238461	63.5354447	1.829506	TP
213	3	1	2.964488917	0.055502202	0.102570482	2.479030243	123.233279	70.2599082	1.753963	TP
214	3	1	0.928228052	0.056306359	0.091133492	2.416956655	121.058368	67.5958552	1.790914	TP
215	3	1	4.071914525	0.075224774	0.070256815	2.455400183	125.617436	74.2770095	1.691202	TP
216	3	1	2.819422499	0.078870068	0.026193613	2.443676943	125.08978	77.913226	1.605501	TP
217	3	1	2.567748639	0.076375023	0.031070982	2.508236166	122.882672	76.7095304	1.601922	TP
218	3	1	2.21963455	0.096146922	0.031628684	2.483377497	118.831895	78.9512681	1.50513	TP
219	2	1	2.569073317	0.099515063	0.022649041	2.480292472	117.972464	84.2410353	1.400416	FP
220	2	1	3.515951088	0.132953674	0.024699653	2.444853462	111.479574	88.8902873	1.254125	FP
221	2	0	9.076697477	0.203732067	0.163997369	2.072320359	117.005254	62.9007608	1.860156	TN
222	2	1	6.813488891	0.259348771	0.157126299	2.297681193	107.191523	98.1430845	1.092196	FP
223	2	1	6.07779895	0.276752267	0.811826729	0.326211731	105.103144	92.2738956	1.139034	FP
224	2	1	0.899855109	0.288243376	0.949834256	0.370142551	103.777654	92.2463634	1.125005	FP
225	2	1	0.822768788	0.299599573	0.850987295	0.741449563	103.448789	94.5763642	1.093812	FP

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
226	2	1	2.355160247	0.096636645	0.722379287	0.756687139	118.836836	88.0357172	1.349871	FP
227	2	1	4.635339647	0.092859744	0.235585439	0.92893131	106.673323	86.7998924	1.228957	FP
228	2	1	2.511649601	0.094068661	0.203294307	0.924296646	107.737413	84.7972781	1.270529	FP
229	2	0	4.608179798	0.101589853	0.110080937	1.028037712	107.644793	77.8724108	1.382323	TN
230	4	0	7.835892522	0.054818276	0.254321985	1.506772561	101.329471	76.829095	1.318895	TN
231	4	0	4.9089108	0.112061965	0.291381325	1.589163624	106.71199	68.8675672	1.549525	TN
232	4	0	3.861365502	0.193977848	0.319122914	1.72771992	120.304558	66.6838348	1.804104	TN
233	1	0	2.65437138	0.260103276	0.268076664	1.783515459	129.145591	64.2354601	2.010503	TN
234	1	0	5.675145459	0.271500838	0.147859479	1.928476506	136.618623	67.6766262	2.018697	TN
235	1	0	3.667852515	0.23568269	0.124614003	1.909328237	139.270674	61.8797564	2.250666	TN
236	1	0	8.473087245	0.359292488	0.08980198	1.715193494	133.604576	46.9317042	2.846787	TN
237	1	0	14.88570833	0.351929321	0.079860992	1.806174709	147.1097	54.0123985	2.723628	TN
238	1	0	8.899193949	0.506351402	0.088094802	1.731117513	147.231118	42.3685469	3.47501	TN
239	1	0	11.11612027	0.421383868	0.068821667	1.802645788	149.031793	46.2998875	3.218837	TN
240	1	0	4.621001438	0.274462408	0.037016826	1.75174754	147.435206	50.7504295	2.905103	TN
241	1	0	7.958353237	0.296237086	0.031776296	1.740068576	143.265886	52.7326246	2.716836	TN
242	1	0	14.01057889	0.26254951	0.032543163	1.809180382	130.100019	43.5669214	2.986211	TN
243	1	0	9.502812029	0.163898117	0.065577578	1.626696209	144.004184	47.4949526	3.031989	TN
244	1	0	8.305967894	0.201071444	0.076169956	1.611982734	138.836225	41.6365497	3.33448	TN
245	1	0	7.548528552	0.304430141	0.078760927	1.622765059	134.037027	37.2350951	3.59975	TN
246	1	0	6.224021295	0.232847215	0.076573504	1.611799774	128.88635	37.7394107	3.415166	TN
247	1	0	3.104266294	0.18329665	0.009233994	1.600501543	126.985409	38.0737852	3.335245	TN
248	1	0	1.610854798	0.138090291	0.007060941	1.610454104	125.65332	39.5772314	3.174889	TN
249	1	0	3.443251448	0.223147504	0.01110606	1.632819832	120.622053	41.0202017	2.940552	TN
250	1	0	2.836004922	0.212067593	0.011739752	1.626900793	118.674024	41.2815611	2.874747	TN
251	1	0	2.616513444	0.168340723	0.011649614	1.622281221	119.881829	40.0278533	2.99496	TN
252	1	0	5.277549996	0.127803951	0.021015748	1.573884946	119.271062	37.3195079	3.195944	TN
253	1	0	2.109286948	0.253951222	0.030915944	1.557176359	123.171575	34.4352605	3.576903	TN
254	1	0	4.05291216	0.32840361	0.034640246	1.540731405	121.762193	32.6806954	3.725814	TN
255	1	0	1.145420068	0.340253609	0.032812341	1.527921714	123.932032	31.6196168	3.919467	TN
256	1	0	2.359116675	0.259407939	0.016020636	1.539441949	121.418075	31.4033557	3.866404	TN
257	1	0	6.46129139	0.176010323	0.027810453	1.606818762	113.215961	32.8350085	3.448026	TN
258	1	0	3.338090425	0.222035715	0.041793688	1.631798662	111.22829	33.0611166	3.364323	TN
259	1	0	4.258674036	0.253235752	0.054602246	1.671341913	111.8555	33.5326565	3.335718	TN
260	1	0	6.359563493	0.235672149	0.058872569	1.71313116	110.792493	35.080057	3.158276	TN
261	1	0	5.988629721	0.117091141	0.047821062	1.734279265	105.497638	33.4541876	3.153496	TN
262	1	0	7.287506224	0.295295962	0.036047685	1.668655024	89.9470832	35.3050094	2.547714	TN
263	1	0	4.126264196	0.354751627	0.038994989	1.621923535	82.9139275	33.6398351	2.464754	TN
264	1	0	1.895489022	0.311861602	0.052968103	1.593919216	82.4862251	32.2791257	2.555405	TN
265	1	0	2.676535871	0.246485301	0.052218705	1.600636843	81.7055086	30.294511	2.69704	TN
266	1	0	0.82049239	0.077076164	0.027478278	1.60019695	81.8089378	31.2949906	2.614122	TN
267	1	0	0.445989006	0.079992327	0.00951161	1.606353956	82.2025834	31.0570033	2.646829	TN
268	1	0	0.972745427	0.073244471	0.004603966	1.606244971	83.5406952	30.1500688	2.770829	TN
269	1	0	0.96266024	0.070993569	0.024249213	1.663622332	84.6717716	30.2319846	2.800735	TN
270	1	0	3.82393891	0.23964826	0.048345155	1.725282431	81.6381559	38.2238074	2.135793	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
271	1	0	0.658626793	0.325263821	0.053556448	1.726775683	80.1084339	39.3044928	2.03815	TN
272	1	0	0.252613308	0.325971358	0.049114766	1.731907669	82.7406485	37.2841646	2.21919	TN
273	1	0	1.414296963	0.267177518	0.025954724	1.729102514	82.3788478	36.9978248	2.226586	TN
274	1	0	2.353616491	0.087696659	0.002397252	1.726131118	80.1230459	39.7208301	2.017154	TN
275	1	0	1.738793766	0.137211293	0.002631054	1.732670585	84.120447	35.2071403	2.389301	TN
276	1	0	0.924562343	0.119036944	0.002303813	1.729907179	82.5485285	37.9618598	2.174512	TN
277	1	0	0.820100298	0.121473418	0.016734696	1.687941806	82.7877584	36.5536065	2.264831	TN
278	1	0	2.482130116	0.121469121	0.022994173	1.759820547	84.218084	37.8275466	2.226369	TN
279	1	0	2.368100782	0.07955623	0.023013318	1.730345243	86.0660611	39.5984868	2.173468	TN
280	1	0	2.097546259	0.075027058	0.024212102	1.746649189	90.4099205	38.0406247	2.376668	TN
281	1	0	0.528675428	0.089527805	0.025921084	1.754365109	92.3311195	38.3180983	2.409596	TN
282	1	0	10.81708576	0.149821088	0.033995729	1.829073562	70.509278	35.351265	1.994533	TN
283	1	0	6.639466177	0.193122965	0.057588713	1.648778361	91.6571955	36.0363787	2.543463	TN
284	1	0	4.181042519	0.268901145	0.078188124	1.611722936	94.9213887	33.6042779	2.824682	TN
285	1	0	3.279600074	0.3324113	0.087871636	1.604579188	94.2549036	32.1019318	2.936113	TN
286	1	0	2.624342645	0.393566228	0.090875987	1.574093465	93.8391426	30.0119015	3.126731	TN
287	1	0	6.910035184	0.453747177	0.023874352	1.604236984	114.450504	29.4279807	3.889173	TN
288	1	0	4.163877353	0.510375414	0.012995725	1.596380568	120.287132	29.5646041	4.06862	TN
289	1	0	4.040726486	0.438664107	0.014944096	1.569805453	115.462327	31.2118619	3.699309	TN
290	1	0	2.925051685	0.323403586	0.017020644	1.558647177	113.959683	32.3063303	3.527472	TN
291	1	0	3.414109044	0.278342384	0.017438231	1.569747546	114.279535	34.9662824	3.268278	TN
292	1	0	4.552720213	0.340707606	0.016782595	1.601852111	116.270725	37.6215643	3.090534	TN
293	1	0	6.053289377	0.313861699	0.029534036	1.639416826	120.021555	42.6888248	2.811545	TN
294	1	0	3.063532338	0.269274853	0.029876757	1.617684298	121.213516	42.7056243	2.83835	TN
295	1	0	4.618504286	0.212439259	0.029013889	1.562037894	118.136537	44.1068283	2.678418	TN
296	1	0	7.10538167	0.302162677	0.034249221	1.547539137	99.0369968	45.4973158	2.176766	TN
297	1	0	3.180509581	0.246174223	0.045692554	1.515372836	119.314857	42.9175099	2.780097	TN
298	1	0	3.903274929	0.233782258	0.053324767	1.456130431	120.812578	45.9400929	2.629785	TN
299	1	0	3.516010576	0.21070698	0.043450983	1.460962506	123.804117	46.4400552	2.665891	TN
300	1	0	4.657403848	0.205924358	0.034375883	1.50392443	120.637382	46.5411416	2.592059	TN
301	1	0	5.572520841	0.066192093	0.033939765	1.546054185	120.475664	44.3192661	2.718359	TN
302	1	0	6.619888737	0.151851468	0.054881958	1.602144309	129.253402	42.8476786	3.016579	TN
303	1	0	4.50094782	0.269434523	0.056282975	1.606869785	132.93173	40.0746247	3.317105	TN
304	1	0	5.1332479	0.419234518	0.038380161	1.578140788	140.365803	37.4250612	3.750583	TN
305	1	0	16.37405512	0.341317775	0.029914811	1.635142991	139.186399	43.4302566	3.204826	TN
306	1	0	4.082636621	0.253036144	0.021682896	1.635373625	131.837223	42.1136089	3.130514	TN
307	1	0	5.958228683	0.378227076	0.022388574	1.632106649	127.383602	49.5164326	2.572552	TN
308	1	0	7.164118141	0.392347864	0.022004371	1.611773432	137.442498	48.0143864	2.862527	TN
309	1	0	7.783832466	0.345036992	0.011785984	1.648297663	122.573874	53.6846814	2.283219	TN
310	1	0	11.04383938	0.315201065	0.027960206	1.695358261	133.248885	56.3079959	2.366429	TN
311	1	0	10.44376658	0.200414686	0.029995577	1.676375495	131.333898	52.8696976	2.484105	TN
312	1	0	8.772979117	0.199018253	0.029143852	1.676742852	133.625914	52.5504242	2.542813	TN
313	1	0	7.068251276	0.106881794	0.020164045	1.707793079	126.918972	55.7595948	2.276182	TN
314	1	0	5.532184711	0.180231613	0.011938875	1.685625262	117.313859	57.7738296	2.030571	TN
315	1	0	4.106881092	0.296741992	0.028134329	1.750917786	111.97605	64.2377904	1.743149	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
316	1	0	1.625751795	0.311313513	0.032124623	1.753619472	113.741617	65.4429068	1.738028	TN
317	1	0	1.838222829	0.215769233	0.025831899	1.72502049	115.833485	66.3300867	1.746319	TN
318	1	0	1.29666144	0.132785991	0.03638644	1.661376172	114.342362	69.9498471	1.634633	TN
319	1	0	2.726541495	0.071035036	0.089331525	1.515422248	108.286084	68.8818814	1.572055	TN
320	1	0	4.857916481	0.066013593	0.088291827	1.583704046	113.857636	69.1222608	1.647192	TN
321	1	0	1.249695571	0.056063777	0.071317281	1.597843402	113.222595	69.128485	1.637857	TN
322	1	0	3.09004067	0.047262081	0.047747427	1.617919551	109.135868	71.5521707	1.525263	TN
323	1	0	4.509873109	0.051021565	0.045157976	1.651715964	112.657792	67.9681798	1.657508	TN
324	1	0	3.040683241	0.049241533	0.038954572	1.691665546	113.662866	69.0135336	1.646965	TN
325	1	0	0.585088959	0.085986622	0.041069692	1.704617275	118.008874	65.717098	1.79571	TN
326	1	0	4.918371674	0.141549516	0.047673391	1.757987121	118.576221	61.1827003	1.938068	TN
327	1	0	6.512863501	0.259678391	0.063981534	1.565976621	111.694707	47.4868054	2.352121	TN
328	1	0	9.72063483	0.32101397	0.063702998	1.655953598	124.336821	50.1423046	2.479679	TN
329	1	0	7.106009719	0.300933507	0.069784773	1.745265798	118.644621	46.5396847	2.549322	TN
330	1	0	7.966339878	0.320303195	0.073444033	1.743098603	131.003945	44.7085384	2.930177	TN
331	1	0	10.09146867	0.217449664	0.093840664	1.844281788	141.193667	49.8913923	2.830021	TN
332	1	0	5.101536535	0.180413186	0.060880627	1.7779907	136.976593	47.8642225	2.861774	TN
333	1	0	9.42987819	0.159297666	0.038667635	1.746165505	132.95137	51.8912889	2.562113	TN
334	1	0	10.40755149	0.128540217	0.044213709	1.715139667	132.889022	48.8263324	2.721667	TN
335	1	0	7.366053112	0.119282823	0.052012748	1.696981838	134.107616	46.458803	2.886592	TN
336	1	0	21.99798827	0.28341599	0.050610532	1.628015719	133.962303	39.3453068	3.404785	TN
337	1	0	3.848462672	0.372934797	0.059381613	1.584067491	135.979006	38.8442183	3.500624	TN
338	1	0	3.546596519	0.297510618	0.057636375	1.573321561	133.945649	41.9023352	3.196615	TN
339	1	0	6.315279269	0.288526579	0.043559472	1.626521718	138.709319	37.0847006	3.740338	TN
340	1	0	2.03609212	0.359132353	0.02354537	1.624005911	130.815919	48.8813181	2.676195	TN
341	1	0	7.598985627	0.391799874	0.021373601	1.610236995	127.618862	44.51296	2.867005	TN
342	1	0	3.98687679	0.393505809	0.027979213	1.557184082	125.867961	46.1290236	2.728607	TN
343	1	0	5.091107921	0.397121332	0.044168115	1.513540992	126.116785	45.5658131	2.767794	TN
344	1	0	4.111287271	0.197054891	0.050266472	1.497964288	135.543619	42.001455	3.227117	TN
345	1	0	3.636126097	0.287222445	0.046975451	1.479277328	136.977831	39.531449	3.465034	TN
346	1	0	0.865636859	0.455948663	0.030991203	1.468827404	138.629594	35.0912246	3.950549	TN
347	1	0	1.115690894	0.554704177	0.017683057	1.468032779	140.335904	32.1930473	4.359199	TN
348	1	0	1.144783517	0.490273711	0.01081614	1.480243886	138.464083	30.8478026	4.488621	TN
349	1	0	1.264089829	0.389581004	0.00608693	1.482468682	136.827712	30.7282364	4.452833	TN
350	1	0	0.447921121	0.210123066	0.010606537	1.49704652	136.182883	30.0794184	4.527444	TN
351	1	0	1.336576405	0.080569774	0.010948419	1.496719796	134.552878	31.2070022	4.311625	TN
352	1	0	0.621276352	0.094339418	0.008539251	1.501417181	135.124049	31.4582827	4.295341	TN
353	1	0	0.430671679	0.094770792	0.008102355	1.506875928	135.741322	31.5511312	4.302265	TN
354	1	0	0.630448611	0.107904246	0.004860078	1.50845152	134.307177	31.9796877	4.199765	TN
355	1	0	1.363231326	0.052142665	0.004442292	1.507321759	134.411273	32.0433679	4.194667	TN
356	1	0	0.424380018	0.060261583	0.003897391	1.513634248	134.787872	32.4847639	4.149264	TN
357	1	0	0.771683721	0.051577421	0.002648914	1.511809148	134.897617	31.7968323	4.242486	TN
358	1	0	0.117334736	0.069115872	0.004294597	1.519423951	135.604784	31.1536678	4.352771	TN
359	1	0	0.127271111	0.069924016	0.005448344	1.522606386	134.971817	31.5448789	4.278724	TN
360	1	0	0.348166563	0.065939388	0.004490103	1.522491379	135.13325	31.8938493	4.236969	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
361	1	0	0.343452307	0.068143687	0.004590427	1.525030784	134.520607	32.4751274	4.142266	TN
362	1	0	0.287035209	0.073576334	0.003735461	1.530599403	134.621062	32.1862171	4.182569	TN
363	1	0	0.848778083	0.055092851	0.003359131	1.529240076	134.316861	32.4743414	4.136092	TN
364	1	0	0.448913178	0.040106254	0.003025703	1.528990165	134.633596	32.5965429	4.130303	TN
365	1	0	1.320067927	0.01846543	0.002230318	1.525373682	134.729213	32.5232899	4.142546	TN
366	1	0	0.556288826	0.019086948	0.00528293	1.516071461	136.02074	32.8984834	4.13456	TN
367	1	0	1.45668539	0.029607324	0.006700975	1.513151115	133.36245	32.8275789	4.062513	TN
368	1	0	0.373309048	0.031726256	0.007971682	1.507253199	133.489972	32.1544872	4.151519	TN
369	1	0	0.284015486	0.041019548	0.006394889	1.50902657	134.427018	32.0964562	4.18822	TN
370	1	0	0.559624364	0.066226377	0.003447641	1.507548797	134.372398	31.5098703	4.264454	TN
371	1	0	0.309977902	0.076943905	0.003608418	1.501917439	134.45507	31.4902385	4.269738	TN
372	1	0	0.447269328	0.045634586	0.002473926	1.505036963	134.665786	31.7838343	4.236927	TN
373	1	0	1.505774908	0.030472185	0.002621349	1.503318509	134.908488	31.6384267	4.264071	TN
374	1	0	0.513254963	0.012241854	0.001880466	1.504232759	135.531849	31.7442573	4.269492	TN
375	1	0	1.373024773	0.031039632	0.0013222	1.505678355	136.96282	32.6988523	4.188612	TN
376	1	0	0.122565745	0.043164688	0.000860385	1.503718938	137.89604	33.1558787	4.159022	TN
377	1	0	0.201868438	0.069867658	0.000944267	1.502977842	137.089251	33.5842643	4.081949	TN
378	1	0	0.326577047	0.065781221	0.003231141	1.511922019	138.340525	33.6790792	4.10761	TN
379	1	0	0.942641958	0.037783901	0.005374248	1.516967195	138.669444	33.4404908	4.146753	TN
380	1	0	1.356784585	0.027908913	0.009667934	1.529244016	137.837569	33.3283667	4.135743	TN
381	1	0	1.930706804	0.041979684	0.012270326	1.537423449	136.707109	32.4991006	4.206489	TN
382	1	0	1.82740671	0.057726712	0.009910092	1.534381554	134.430231	33.361966	4.029446	TN
383	1	0	2.239301125	0.063412507	0.007019044	1.527705583	132.590285	32.6544317	4.060407	TN
384	1	0	1.561871501	0.065426055	0.005680953	1.520986787	130.988744	32.3257332	4.052151	TN
385	1	0	1.062529979	0.064314033	0.007452974	1.518031579	131.68237	31.9201978	4.125362	TN
386	1	0	1.441730117	0.049503982	0.005888527	1.521141047	131.622745	31.6284421	4.161531	TN
387	1	0	1.385251135	0.060784877	0.009807174	1.545172966	132.467208	31.4446532	4.21271	TN
388	1	0	1.441742199	0.052756195	0.014241223	1.55218522	130.736675	31.7304582	4.120227	TN
389	1	0	0.019740793	0.066574703	0.01789746	1.564252987	131.60298	30.6093286	4.29944	TN
390	1	0	0.552598005	0.060042225	0.016852067	1.568879381	131.910641	31.3196406	4.211755	TN
391	1	0	1.430446842	0.056694373	0.009088417	1.549125505	129.944136	30.837538	4.21383	TN
392	1	0	3.221409199	0.07862192	0.008043666	1.550321784	131.271458	30.1947073	4.347499	TN
393	1	0	3.818385586	0.121897018	0.007924166	1.553287753	141.378894	31.1132362	4.544011	TN
394	1	0	3.55810903	0.125167239	0.007732063	1.547689077	140.670623	33.0818152	4.252204	TN
395	1	0	3.568390836	0.13386796	0.001976983	1.548313926	139.740794	33.5504142	4.165099	TN
396	1	0	6.586540221	0.383877806	0.004139825	1.540776785	132.178885	38.6416045	3.420637	TN
397	1	0	3.983595262	0.42692931	0.008744012	1.566974915	132.694282	37.2239357	3.564757	TN
398	1	0	4.510510431	0.381539185	0.013479722	1.576693295	130.915387	39.0872408	3.349313	TN
399	1	0	9.988740199	0.321246378	0.012999486	1.563738339	139.518993	42.7144066	3.266322	TN
400	1	0	8.035475907	0.164425909	0.014269465	1.541921	147.676259	48.1004694	3.070163	TN
401	1	0	13.34008053	0.231044349	0.042919182	1.458867578	139.740613	37.4384012	3.732548	TN
402	1	0	8.641332799	0.220792875	0.041204673	1.545381553	135.109067	41.8137757	3.231209	TN
403	1	0	10.86676416	0.353918915	0.113442825	1.796259168	146.80726	55.7539361	2.633128	TN
404	1	0	11.9774773	0.358571189	0.125437885	1.721205744	147.953162	44.2738466	3.341773	TN
405	1	0	6.784402193	0.390092637	0.123834281	1.703471252	152.206661	41.6723483	3.652462	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
406	1	0	18.102101	0.437790231	0.084588858	1.746642727	105.224255	42.1404264	2.496991	TN
407	1	0	26.18961612	0.607197613	0.032458354	1.719949046	126.231445	64.3922457	1.960352	TN
408	1	0	17.93868688	0.880640959	0.113588731	1.44095008	105.383044	84.1834256	1.251827	TN
409	1	0	2.975467544	0.889329573	0.141648537	1.430384966	105.715312	81.8136582	1.292147	TN
410	1	0	3.686006487	0.497018397	0.162369556	1.353114482	108.403616	84.5249599	1.282504	TN
411	1	0	2.087753606	0.265670393	0.148809169	1.28236072	112.494969	76.8330248	1.464149	TN
412	1	0	1.429428347	0.152084076	0.114718463	1.128369861	120.252102	72.6841609	1.654447	TN
413	5	0	0.9113539	0.208082772	0.167073989	0.963170648	128.573284	70.7379644	1.817599	TN
414	5	1	4.674938184	0.218125173	0.189907663	0.845952784	126.618607	67.9290542	1.863983	FP
415	5	1	3.42431972	0.222408903	0.184840559	0.777428	141.805786	66.5293324	2.131478	FP
416	5	1	3.474400887	0.176321008	0.146259362	0.716090929	142.677041	68.4484416	2.084445	FP
417	5	1	3.221921119	0.135348939	0.100475881	0.680641925	134.900786	73.9494091	1.824231	FP
418	5	1	3.745036138	0.145936484	0.083476971	0.600822526	127.777611	72.3152181	1.766953	FP
419	5	1	2.58395848	0.142483538	0.05753813	0.676255845	142.68594	74.5773041	1.913262	FP
420	5	1	1.796884435	0.140845009	0.038337387	0.649215198	128.912325	77.2279068	1.669245	FP
421	5	1	1.481560543	0.080515726	0.028622183	0.643013554	134.086487	76.2558586	1.758376	FP
422	5	1	3.469567473	0.098643928	0.025898185	0.665561279	140.482547	72.8995087	1.927071	FP
423	3	1	3.198803686	0.157615786	0.012570892	0.669633964	141.634856	66.5463647	2.128364	TP
424	3	1	7.684339816	0.363682301	0.015971572	0.625503126	155.699841	57.8541379	2.691248	TP
425	3	1	8.710938321	0.315804741	0.051436181	0.52868184	136.93627	62.2239577	2.2007	TP
426	3	1	4.508021556	0.268975769	0.059386264	0.545314315	141.030575	70.6398506	1.996473	TP
427	3	1	2.156073779	0.254993702	0.051929398	0.578801895	141.325922	70.4247455	2.006765	TP
428	3	1	5.323752078	0.327296781	0.060135763	0.444038307	143.017168	52.0902684	2.745564	TP
429	3	1	3.92908969	0.332445279	0.064371543	0.407776098	145.637619	53.6603954	2.714062	TP
430	3	1	5.022769609	0.32621918	0.064556285	0.530764419	146.369851	62.9352441	2.325722	TP
431	3	1	2.200432978	0.273052053	0.069313871	0.574024467	151.912304	60.352768	2.517073	TP
432	3	1	0.57189494	0.208040784	0.065716805	0.559872819	153.556646	52.3455815	2.933517	TP
433	3	1	3.99422722	0.202928454	0.061251221	0.562187445	158.800113	59.537946	2.667208	TP
434	3	1	2.049661063	0.216818951	0.015778772	0.573737214	160.147828	56.6318527	2.827875	TP
435	3	1	0.919448308	0.339095391	0.008751352	0.551059216	167.788829	47.8464473	3.506819	TP
436	3	1	1.008700128	0.365479267	0.012503035	0.536220714	164.496265	46.1829153	3.561842	TP
437	3	1	1.794472073	0.377095007	0.019581057	0.517907838	164.585835	47.674585	3.452276	TP
438	3	1	0.932392557	0.265355766	0.019836582	0.525556519	160.88408	47.9937573	3.352188	TP
439	3	1	1.106532547	0.070234144	0.011375368	0.538570598	161.084945	46.8422403	3.438882	TP
440	3	1	0.934416	0.066746656	0.008734932	0.518154573	162.696368	47.006548	3.461143	TP
441	3	1	0.840569337	0.039602235	0.01533981	0.491601	162.387799	47.1296586	3.445554	TP
442	3	1	0.429990767	0.052993582	0.030483006	0.45261313	159.788148	45.4360436	3.516771	TP
443	3	1	1.465748714	0.074748605	0.037654121	0.439447834	156.755768	47.6211753	3.291724	TP
444	3	1	0.856616582	0.078073649	0.028172275	0.466359386	162.081745	48.0733929	3.371548	TP
445	3	1	1.756991538	0.084360865	0.017392032	0.46768924	162.486448	46.3816547	3.503248	TP
446	3	1	2.647101842	0.129337919	0.010475694	0.461578766	168.009354	45.8145249	3.667164	TP
447	3	1	1.829652798	0.134809238	0.010558118	0.451788269	162.604589	45.5482179	3.569944	TP
448	3	1	2.309389532	0.096360662	0.006006806	0.467406549	164.745325	46.4827735	3.544223	TP
449	3	1	1.099990849	0.054023935	0.006313475	0.468567282	168.536336	47.2841801	3.564328	TP
450	3	1	1.145753364	0.047038593	0.0085654	0.477765018	166.62262	45.8200166	3.636459	TP

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
451	3	1	0.474688895	0.031186224	0.009214947	0.476141453	165.755549	46.2097657	3.587024	TP
452	3	1	0.736649724	0.032103817	0.004243572	0.469377141	165.532243	45.9032758	3.60611	TP
453	3	1	0.636544462	0.03444345	0.003853537	0.469626168	166.406386	45.4529911	3.661066	TP
454	3	1	0.722980107	0.026383676	0.003381878	0.473850367	165.936616	45.5734809	3.641078	TP
455	3	1	0.228572838	0.036466033	0.002749412	0.474737239	165.711723	44.9325773	3.688008	TP
456	3	1	0.153572527	0.029933051	0.002249268	0.470353759	167.143671	45.3849515	3.682799	TP
457	3	1	0.408138115	0.020339794	0.007561613	0.453883017	163.541521	44.9393929	3.639157	TP
458	3	1	1.058497313	0.022209125	0.007891745	0.474028631	167.1011	45.3448039	3.685121	TP
459	3	1	2.151706759	0.043808469	0.007771788	0.463821982	164.467276	46.0194186	3.573867	TP
460	3	1	3.058921731	0.066726239	0.006875588	0.46721886	167.784775	44.4094207	3.778135	TP
461	3	1	2.638277468	0.090836194	0.007253181	0.472792854	162.786372	46.3201732	3.514373	TP
462	3	1	1.229402715	0.09087667	0.00421041	0.474446502	166.188241	45.5963034	3.644774	TP
463	3	1	1.136691236	0.088630515	0.005638242	0.459188922	163.946323	44.849401	3.655485	TP
464	3	1	0.992672072	0.084695728	0.005337844	0.467734451	165.524723	45.8178167	3.612672	TP
465	3	1	1.450345766	0.101310451	0.005661082	0.463644999	161.349518	47.6486948	3.386232	TP
466	3	1	0.985272	0.099336436	0.005637007	0.472661545	162.228056	45.8204611	3.540516	TP
467	3	1	0.883626962	0.09709661	0.004498507	0.464222493	164.769127	45.4009974	3.629196	TP
468	3	1	2.969763035	0.127454176	0.00331778	0.469155305	167.802585	44.4200676	3.77763	TP
469	3	1	1.535731551	0.138450844	0.003535808	0.464349051	166.681669	44.7862626	3.721714	TP
470	3	1	1.226336972	0.087664104	0.003173632	0.468254085	166.184214	46.389419	3.582373	TP
471	3	1	2.966597828	0.080106406	0.004749024	0.477267024	168.90654	44.6589384	3.782144	TP
472	3	1	1.758409165	0.122307019	0.009100968	0.489939	167.429217	48.2622708	3.469153	TP
473	3	1	2.282736947	0.109010987	0.011765662	0.494474712	165.920639	45.519529	3.645043	TP
474	3	1	0.405629206	0.138776851	0.021169811	0.530072346	163.777348	48.4267209	3.381962	TP
475	3	1	1.816226792	0.193317057	0.023495291	0.537057635	160.047105	49.5061078	3.232876	TP
476	3	1	0.261311414	0.152187751	0.026990814	0.561585786	160.560064	49.4191846	3.248942	TP
477	3	1	2.053310433	0.150124374	0.02158893	0.536209506	163.309943	49.3426755	3.30971	TP
478	3	1	2.363929248	0.066874647	0.030231164	0.470625448	161.066641	50.4971674	3.189617	TP
479	3	1	1.13194461	0.120107399	0.041188222	0.456336564	168.248138	47.6664366	3.529698	TP
480	3	1	1.990537737	0.137168459	0.046310701	0.444918015	166.030541	47.3506621	3.506404	TP
481	3	1	0.883045803	0.149897979	0.035623908	0.436054647	166.749463	46.4943836	3.586443	TP
482	3	1	2.538665228	0.140164073	0.011593664	0.450424673	165.379131	48.6521556	3.399215	TP
483	3	1	1.292600676	0.314101519	0.013602175	0.476557752	155.306709	56.7849347	2.734998	TP
484	3	1	9.109731453	0.316692313	0.026274155	0.395622969	155.747449	43.853534	3.551537	TP
485	3	1	6.240394747	0.540442433	0.046564534	0.535917919	140.196104	63.519487	2.207135	TP
486	3	1	6.890165142	0.483400102	0.076531459	0.310126295	142.799805	49.7869734	2.868216	TP
487	3	1	3.240701584	0.440446863	0.104973972	0.248722242	134.924839	52.1522414	2.587134	TP
488	3	1	1.322173897	0.482422569	0.108590706	0.247556139	130.644237	56.6366214	2.30671	TP
489	3	1	5.420031514	0.231142976	0.13833434	0.113076984	130.497688	53.1387326	2.455792	TP
490	3	1	3.949805008	0.233248387	0.075130216	0.133221187	134.628339	61.2379263	2.198447	TP
491	3	1	31.11010598	0.50874044	0.910960825	2.458698808	66.2646301	57.1958532	1.158557	TP
492	3	1	28.11444406	0.493505552	0.924340075	0.111243961	139.478473	55.9330197	2.49367	TP
493	3	1	2.145291778	0.516629733	0.929145511	0.190189112	127.070071	50.3719337	2.522636	TP
494	3	1	4.887045248	0.496002794	0.914800046	0.26826439	123.404711	59.7760397	2.064451	TP
495	3	1	5.911552946	0.496315748	0.904916033	0.230971232	125.374086	56.8517856	2.20528	TP

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
496	3	1	2.504865085	0.173142273	0.067585255	0.307671899	126.054243	54.6815745	2.305242	TP
497	3	1	1.096310042	0.149284067	0.046516939	0.312738352	120.584537	53.01958	2.27434	TP
498	3	1	3.374965142	0.101989071	0.032506873	0.313535065	123.669264	52.3865232	2.360708	TP
499	3	1	4.56190005	0.17195109	0.039571973	0.352140188	137.489638	50.9685476	2.697539	TP
500	3	1	1.631216545	0.158279316	0.025493494	0.371262566	136.170203	53.9176875	2.52552	TP
501	3	1	2.343514149	0.216984003	0.047482345	0.441889065	137.829755	48.0522615	2.86833	TP
502	3	1	1.895732023	0.205200192	0.05366816	0.454299918	135.796744	46.7912128	2.902185	TP
503	3	1	8.678598321	0.197751728	0.074377164	0.562759403	149.660943	48.1057786	3.11108	TP
504	3	1	1.315980076	0.324061442	0.071440046	0.549395218	154.525915	44.0009628	3.511876	TP
505	3	1	2.152448923	0.231745377	0.052295347	0.550140712	152.180227	47.7835456	3.184783	TP
506	3	1	0.803951737	0.218571461	0.040545409	0.557367724	154.096653	45.0661798	3.419341	TP
507	3	1	1.02313933	0.161914751	0.006448216	0.544482339	154.506187	49.2591124	3.136601	TP
508	3	1	0.937341446	0.172816131	0.008275905	0.532394488	153.161618	50.0383737	3.060883	TP
509	3	1	0.985341474	0.135587077	0.00887471	0.554760582	154.067241	45.8625769	3.359324	TP
510	3	1	2.415278714	0.183496721	0.01406085	0.5198329	147.190986	50.2477384	2.929306	TP
511	3	1	1.651299113	0.1684173	0.011794025	0.541602916	153.87012	45.833312	3.357168	TP
512	3	1	0.217052353	0.173467658	0.01265476	0.550760991	149.63711	48.8160169	3.065328	TP
513	3	1	2.85863736	0.293453649	0.015201939	0.564778476	150.742587	58.5496708	2.57461	TP
514	3	1	3.169837698	0.31936928	0.017953442	0.570335425	149.242792	59.9432285	2.489736	TP
515	3	1	9.285795234	0.382157526	0.017006577	0.522740052	134.708652	57.4727342	2.343871	TP
516	3	1	8.057796251	0.333280795	0.055128384	0.683676697	122.572637	59.8987196	2.046332	TP
517	3	1	2.08913725	0.210288669	0.110261264	0.826606018	126.69067	60.5990304	2.090639	TP
518	3	0	3.432018189	0.193161989	0.172820507	0.994482011	117.937438	59.4511501	1.983771	FN
519	2	0	1.281235498	0.184591226	0.20356714	1.084178103	111.74165	63.1054696	1.770713	TN
520	2	0	1.758015472	0.18158107	0.149974067	1.044971946	111.737078	69.3476704	1.611259	TN
521	2	0	0.430974764	0.226231369	0.097447552	1.093004051	109.842115	74.1841312	1.480669	TN
522	2	0	4.601829036	0.197121382	0.037620429	1.018440442	111.926155	77.0073912	1.453447	TN
523	2	1	4.275003049	0.112463086	0.118131222	0.77259836	122.644192	77.4754201	1.583008	FP
524	2	0	5.278523473	0.059494638	0.123386381	1.117609411	109.40759	71.2564358	1.535406	TN
525	4	0	5.465722565	0.04537155	0.155904317	1.242792365	107.92519	72.247093	1.493834	TN
526	4	0	4.146795715	0.045717606	0.259647293	1.559858091	111.595186	71.7729872	1.554835	TN
527	4	0	5.565755591	0.047425362	0.310118633	1.624529675	120.143169	73.4571577	1.635554	TN
528	4	0	3.870038069	0.204555938	0.234220758	1.730130514	133.177871	64.8649398	2.053156	TN
529	1	0	7.352372061	0.233660004	0.211471693	1.877672147	127.918673	63.8458768	2.003554	TN
530	1	0	12.38758821	0.212986096	0.123957518	1.852286926	124.509897	61.6373392	2.02004	TN
531	1	0	9.289940149	0.313318216	0.091210639	1.746491665	128.410888	49.1400563	2.613161	TN
532	1	0	12.52372245	0.525422898	0.058268599	1.77727309	142.805149	42.5551715	3.355765	TN
533	1	0	11.24154277	0.585645888	0.054324617	1.74890535	135.092916	41.1477911	3.283115	TN
534	1	0	11.14225417	0.49559648	0.038357074	1.788305673	141.598924	46.3929967	3.052162	TN
535	1	0	10.08770657	0.375485994	0.07344835	1.944402252	134.639486	56.1723286	2.396901	TN
536	1	0	12.66480207	0.352779113	0.083060667	1.696155526	140.630907	50.7796962	2.769432	TN
537	1	0	9.349035282	0.303467759	0.109470051	1.615351052	135.293245	49.9865022	2.706596	TN
538	1	0	8.868529358	0.212689388	0.134079911	1.567124783	136.048891	47.9155286	2.839349	TN
539	1	0	9.106709183	0.240721453	0.139674557	1.577749432	132.171791	42.0066066	3.146452	TN
540	1	0	7.997666272	0.176177949	0.046156337	1.592034831	127.16948	41.1271526	3.092105	TN

Image No.	Label	Fall Flag	Centroid Distance	Axis Ratio std. dev.	Angle std. dev.	Orientation Angle	Maj. Axis Length	Min. Axis Length	Axis Ratio	Result
541	1	0	4.711701495	0.161382008	0.016229014	1.591891362	123.376024	41.9088811	2.943911	TN
542	1	0	4.188424455	0.123521096	0.009523244	1.586636429	122.701603	42.9406642	2.857469	TN
543	1	0	6.911886788	0.112753347	0.005527701	1.591797863	118.34922	40.8802063	2.895025	TN
544	1	0	5.875727555	0.080734193	0.006088187	1.576252261	116.38952	39.0712509	2.978904	TN
545	1	0	4.179411879	0.047756391	0.016646512	1.547538112	114.282057	39.9683497	2.859314	TN
546	1	0	4.999652526	0.116804072	0.029068206	1.51378297	111.992522	42.6279389	2.627209	TN
547	1	0	5.518133277	0.116896588	0.049330973	1.453483432	111.428154	38.9472471	2.861002	TN
548	1	0	7.885595252	0.116056156	0.041908716	1.499674805	105.823225	36.6867877	2.884505	TN
549	1	0	6.778478123	0.192491125	0.030343318	1.510887761	106.899565	33.1137718	3.228251	TN
550	1	0	5.479907419	0.28633915	0.02367297	1.518240787	104.281081	30.3868758	3.43178	TN
551	1	0	2.512398319	0.343722191	0.02410132	1.517008842	107.432923	28.4755849	3.772808	TN
552	1	0	3.496839551	0.323518371	0.008902963	1.526466664	105.325166	28.4903556	3.696871	TN
553	1	0	3.291549761	0.214024405	0.006707723	1.52942877	102.825794	27.3448291	3.760338	TN
554	1	0	239.4838106	0.293385933	0.019777163	1.570796327	0	0	#DIV/0!	TN
555	1	0	0	0.365069882	0.02314429	1.570796327	0	0	#DIV/0!	TN
556	1	0	0	0.357505561	0.021012335	1.570796327	0	0	#DIV/0!	TN
557	1	0	0	0.304135109	0.016547023	1.570796327	0	0	#DIV/0!	TN
558	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
559	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
560	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
561	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
562	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
563	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
564	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
565	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
566	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
567	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
568	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
569	6	0	0	0	0	1.570796327	0	0	#DIV/0!	TN
570	6	1	199.0418938	0.632443774	0.62560456	3.134807727	377.096876	265.768824	1.418891	FP