FALL DETECTION SYSTEM

Justin Haryanto

Nhan Nguyen

CONCEPT OF OPERATIONS

REVISION – 2.0

30 April 2023

Concept of Operations

for

FALL DETECTION

Team <56>

Approved by:

____

Project Leader Date

____

Prof. Kalafatis Date

____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	15 Sep 22	Justin Haryanto		Draft Release
2.0	29 Apr 23	Justin Haryanto		Revision

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

No table of figures entries found.

1. Executive Summary

A fall detection system is beneficial in providing life-saving help to fall victims by notifying emergency services and health personnel as soon as possible. Unfortunately, a lot of current programs require the use of wearables such as smart watches, therefore presenting a financial and physical burden and an inconvenience to those who wish to benefit from such a service.

The proposed system must be able to detect falling motion from humans within video footage at an appropriate frame rate and be able to distinguish between falls and other similar motions. It should also be able to reliably handle interference by the presence of more people and other moving entities. This system utilizes code from pose estimation that have proven successful.

In order to give inputs and receive outputs, a way to interact with the system will need to be developed. Since the detection program is completely online, access will come in the form of a website.

2. Introduction

Training AI to identify human actions is an area of interest for computer scientists with a variety of applications. Action recognition is beneficial in safety, where it is important to handle life-threatening situations quickly. Falls are especially dangerous in areas with an absence of people, or the victim is not treated quickly enough. The objective of this project is to provide an easy to use fall detection system that could potentially interface with existing video recording equipment.

The object of this project is to develop a fall detection system that determines if a fall has occurred after processing given video and image files. Also, a way to interact with the system will need to be developed.

2.1. Background

Fall detection would allow emergency services to help the victim as soon as possible. However, a lot of pre-existing fall detection systems may require extra data from sources such as smart watches which might not always be easily available.

2.2. Overview

Work in the development of fall detection software already exists, however, there is still room to improve speed and accuracy, as well as removing the need for wearables and smart devices. There also needs to be a way to interact with the system that is intuitive, simple, and attractive. With these changes, the safety of the people is both greatly improved and accessible.

2.3. Referenced Documents and Standards

- Fall Detection using Pose Estimation: https://towardsdatascience.com/fall-detection-using-pose-estimation-a8f7fd77081d

- An Overview of Human Pose Estimation with Deep Learning: https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b

- History of Keypoint Detection in Computer Vision: https://medium.com/@SeoJaeDuk/history-of-keypoint-detection-in-computer-vision-be798a32ff4a

- Action Recognition: https://paperswithcode.com/task/action-recognition-in-videos

- What is Action Recognition: https://chooch.ai/computer-vision/what-is-action-recognition

- Action Recognition in Video: https://uwaterloo.ca/vision-image-processing-lab/research-demos/action-recognition-video

- Using Artificial Intelligence to detect and prevent falls: https://hospitalnews.com/using-artificial-intelligence-to-detect-and-prevent-falls/

- Fall Detection: https://viso.ai/application/fall-detection/

- Detecting & Preventing Falls with Artificial Intelligence: https://www.virtusense.ai/blog/detecting-preventing-falls-with-artificial-intelligence

- University of Rzeszow Datasets: http://fenix.univ.rzeszow.pl/mkepski/ds/uf.html

- Bournemouth University: https://falldataset.com/

3. Operating Concept

3.1. Scope

The fall detection system should be able to detect falls of humans when given video and image files. The user-interface should be able to communicate with the system directly and display the results back.

3.2. Operational Description and Constraints

The user uploads videos and image files to the website, which then gets sent to the fall detection system. Pose estimation will process the files by applying keypoint and bounding boxes on people. With this information, the system uses machine learning models to determine if a person has fallen, where it labels each frame where it detects a fall. The results are then sent back to the website and displayed for the user to see.

The main constraint is that test videos do not cover every potential scenario, which makes it more difficult to test the system for flaws.

3.3. System Description

There are two parts to the system: fall detection and externals.

The Fall Detection subsystem contains the pose estimation and fall detection components. Pose estimation focuses on detecting people in the video, then applying keypoint and bounding boxes on them. Fall Detection utilizes machine learning models to determine if a fall has occurred and outputs the result.

The Externals subsystem contains the datasets and website GUI components. Datasets focuses on the collection and creation of datasets for use in training and testing the model. Website GUI focuses on developing a website that allows the user to interact with the fall detection system.

3.3.1. Pose Estimation

Pose estimation identifies people in a video and overlays bounding boxes and key points for use in the fall detection system. Several programs were examined to determine their viability for our project: OpenPose, AlphaPose, and OpenPifPaf.

OpenPose needed training files that were no longer available. Fortunately, those files were found, and it was able to process a video. OpenPifPaf was promising because an author had used it in their fall detection system. However, difficulties running the required programs prevented its use. There was little trouble running AlphaPose, making it the most straightforward of the three.

Their accessibility makes OpenPose and AlphaPose viable candidates for our pose estimation system. Both also have the capability to identify multiple poses at 20 FPS, and their codes are adjustable to work with our fall detection system.

3.3.2. Fall Detection

Fall detection uses the information from the pose estimation system to determine if a person has fallen. This system can distinguish falls from similar motions and can function with poor video quality and unusual poses.

3.3.3. Output

The output can send information to other systems for future processing, but our system only displays a video and text showing which frames a fall was detected.

3.3.4. Datasets

Datasets focuses on collecting and processing datasets used to train and test the fall detection machine learning models.

Training sets were made using preexisting fall datasets from the University of Rzeszow and Bournemouth University, each set containing several hundred images. Only the relevant information was extracted from these sets and processing of the data was performed to make it usable with the system.

On the other hand, testing sets were recorded at the martial arts facility on campus due to the mats the practitioners used, which reduced the risk of injury when performing realistic falls. A total of 78 videos were recorded at 1920x1080 60 FPS, each clip several seconds long. These were compressed down to 480x640 15 FPS, making them more manageable for the system.

3.3.5. Website GUI

The ability to interact with the system using a UI is beneficial in improving the user experience, giving off a finished look, and simplifies the process of testing the system. It was desirable to keep the fall detection system completely online on Google Colab, so several UI solutions were looked at. In the end, using a website as the system's UI was the best option because it was able to upload to the system directly without using Google Drive, offered the most freedom in design, and allowed for communication with the system on any device.

Hosting of the UI was done using Ngrok, which provided the url to access the website. A combination of Flask, HTML, and CSS was used in the development and design of the website.

4. Scenarios

4.1. Senior Care

Falls are a concern for the elderly due to their high likelihood of injury. Implementation of a fall detection system in the camera systems of hospitals and homes would allow medical personnel to react quickly to a fallen individual.

4.2. Isolated Locations

Situations where a person is alone are common, such as when they live by themselves or are walking at night. A fall sustained in this state is dangerous, since seeking immediate help is possibly difficult for many reasons. Security cameras installed with fall detection would improve the safety of the community by automatically contacting emergency services. Such a system is also affordable if the area or house already has such surveillance systems.

5. Analysis

5.1. Summary of Proposed Improvements

This project seeks to create a system to detect falls regardless of body position, improving that ability through the use of datasets, and allow the user to interact with the system through a user interface.

5.2. Disadvantages and Limitations

It is difficult to run the code provided by previous authors, so testing various methods is time-consuming. Another issue is that our group may not have the necessary hardware to run the fall-detection system.

If the system is kept completely on Google Colab, then the challenges that may arise with working in an online environment (such as slow uploads, having a stable connection, and limited affordable software) will need to be dealt with.

5.3. Alternatives

An alternative to fall-detection software is for humans to monitor the cameras, set up patrols around an area, or have every potential victim carry a method of contacting emergency services (such as wearables or apps). Such approaches are expensive, unscalable, and prone to error.

5.4. Impact

Privacy is a possible concern of the public as our system would encourage the use of cameras in the community. As a result, companies would likely attempt to collect information and breach people's privacy.

Another possible impact is the false sense of security that would arise from our program. There is the possibility of our system failing, and excessive trust in it would encourage some people to take more risks. Unless redundancies are put in place, future fall-detection systems may lead to more injuries.

FALL DETECTION SYSTEM

Justin Haryanto

Nhan Nguyen

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 3.0

30 April 2023

Functional System Requirements

for

Fall Detection

Prepared by:

____

Author Date

Approved by:

____

Project Leader Date

____

John Lusher, P.E. Date

____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	03 Oct 2022	Nhan Nguyen		Draft Release
2.0	04 Dec 2022	Justin Haryanto		Revision
3.0	30 Apr 2023	Nhan Nguyen		Revision

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

The purpose of the fall detection system is to provide an accurate and flexible way of detecting falls from videos and camera feeds. The system will be able to operate easily without advanced hardware and be convenient to use.

A website GUI will allow users to upload videos to the system, where the pose estimation system. Users upload images and videos to the system using a website GUI. Once the system receives the files, the pose estimation program identifies people within each image/frame and assigns keypoint and bounding box information to each person. Using machine learning models trained with datasets, the fall detection system will then detect falls and output if a fall has occurred, in the form of a video, to the GUI.

Figure 1 displays the integration of the project in the proposed ConOps.

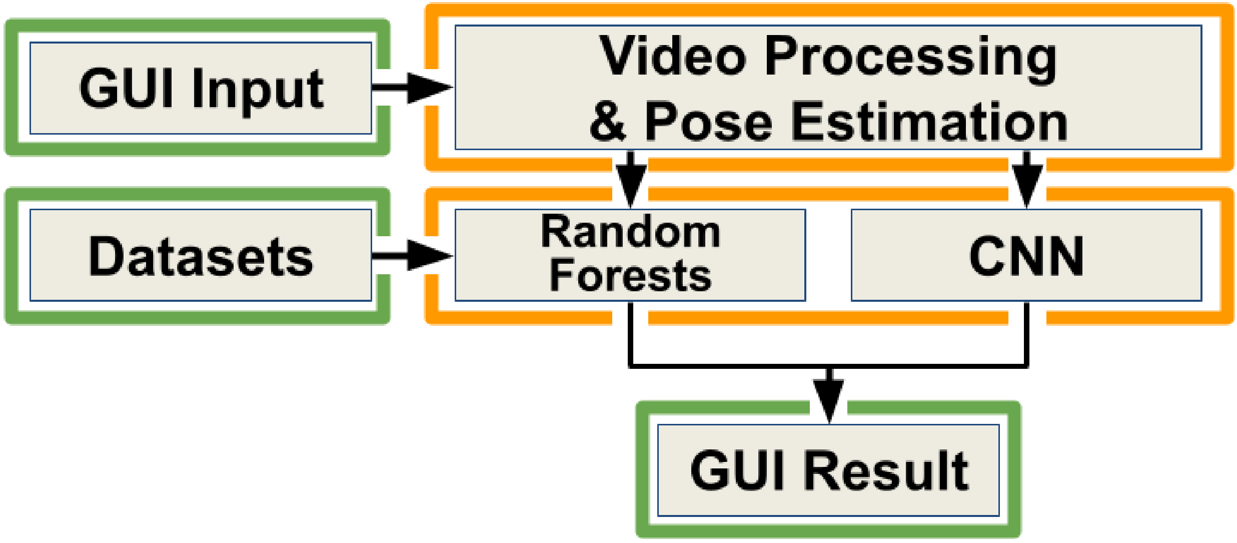


Figure 1: Distribution of Responsibilities

1.2. Responsibility and Change Authority

Changes to performance and implementation requirements can be made by total team agreement. The whole team is responsible for achieving the requirements of the project and their respective subsystems. Responsibility for each subsystem is as follows:

- Nhan Nguyen
    - Video Processing
    - Pose Estimation
    - Fall Detection Machine Learning Models
- Justin Haryanto
    - Datasets Collection
    - System GUI via Website

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Table of Applicable Documents

Document Number	Revision/Release Date	Document Title
N/A	2/4/2018	RMPE: Regional Multi-Person Pose Estimation (AlphaPose)l
N/A	7/20/2017	Fall Detection Data Set
N/A	12/2014	UR Fall Data Set
N/A	4/28/2019	An Overview of Human Pose Estimation with Deep Learning
N/A	1/21/2020	History of Keypoint Detection in Computer Vision
N/A	2.10.0	Tensorflow
N/A	1.12.1	PyTorch
N/A	04/30/2023	https://github.com/Nhannguyen993/TAMU-GitHub
N/A	04/30/2023	https://flask.palletsprojects.com/en/2.2.x/

2.2. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

This section defines the minimum requirements that the development item(s) must meet, as well as the requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system.

3.1. System Definition

The fall detection system comes in two main subsystems: Externals and Fall Detection.

Externals can be decomposed into the Website GUI and Datasets components. Implementing a website GUI allows the user to upload video/image files to the system. On the other hand, Datasets involves the collection and creation, as well as processing, of datasets to use to train and test the system.

The fall detection subsystem uses the AlphaPose pose estimation system to convert the video into a series of images and identifies people within the images. It then applies attributes to their bodies like bounding boxes and key points to be sent to the machine learning models. The fall detection machine learning models determines if a fall has occurred and sends that information back to the website GUI.

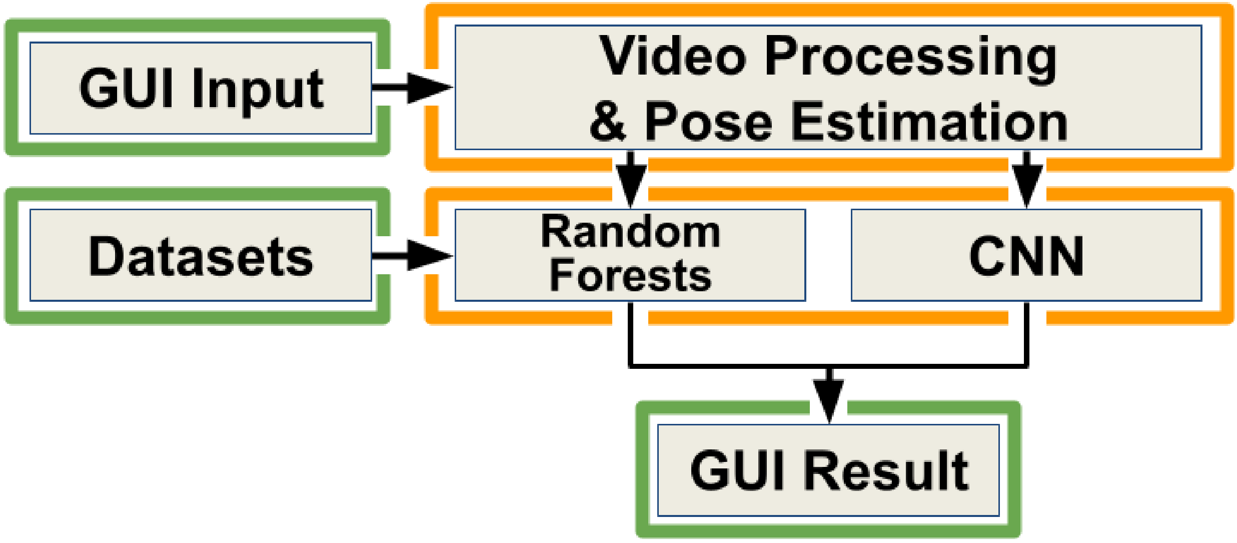


Figure 2: Block Diagram of System

3.1.1. Website GUI

The website GUI allows the user to upload images and videos to the fall detection system. Once the files have been processed, the results are displayed back to the user via the website. UI scaling to the screen size and file extension error checking were implemented to improve the functionality of the interface.

Websites offer the developer full control over the design and functionality of the user-interface. This means that the UI can be made to be attractive and intuitive for the user to use, while also allowing communications with the fall detection system to occur directly.

3.1.2. Datasets

Datasets for the training of the machine learning models were collected from authors who previously researched fall detection, mainly researchers from the University of Rzeszow and Bournemouth University. Both sources have several dozen sets containing over a hundred images, each labeled with the type of motion being performed by the person in the set. (such as standing or falling).

Testing datasets were made by the team, featuring recorded videos of a person falling at the Texas A&M Martial Arts facility. This dataset was used to evaluate the performance of the fall detection system on inputs it has never seen before.

3.1.3. Video Processing & Pose Estimation

Video processing and pose estimation is performed by AlphaPose

During the pose estimation process, AlphaPose places bounding boxes and generates poses on each person. It also outputs files containing keypoint coordinates and videos with the keypoints overlaid, which is used by the fall detection algorithm.

AlphaPose is one of the best pose estimation models because it can handle inaccurate previously placed bounding boxes, a common problem for other systems. This makes it a good pose estimation method for handling videos that will likely have more than one person, which is useful for making the fall detection system more flexible in terms of video inputs. It also has accessible code that doesn’t require significant set up to use.

3.1.4. Fall Detection

Due to its suitability for image classification, CNNs (convolution neural network) are used to analyze images and bounding boxes. On the other hand, the random forest model, a simple classification model, is sufficient for keypoints, thus reducing the need for additional computational resources. Once both models finish processing their respective parts, the results are weighted and combined to give a final verdict.

TensorFlow is a free software library that allows for the building of machine learning models. It is used because it has a framework of the CNN model ready, thus requiring minimal changes to get working.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Website GUI Functionality

Website GUI should be accessible via a typical browser and be able to upload videos or images, but not both at once. It should be able to adjust its layout to match the system displaying it.

> *Rationale: Uploading both images and videos at once complicates the fall detection process (such as the order of the files and if they're from the same source) and should be avoided.*

3.2.1.2. Fall Detection Machine Learning Algorithm Accuracy

The machine learning algorithm used in the fall detection systems should have an accuracy of over 90%.

> *Rationale: An accuracy of over 90% will improve the overall system performance.*

3.2.1.3. Accuracy

The whole system should be able to at least detect over 90% of the falls in a video.

> *Rationale: An accuracy of over 90% allows the system to be comparable to existing fall detection systems.*

3.2.2. Software and Hardware Requirements

This section explains the required hardware and software to run the system.

3.2.2.1. Hardware

Computers utilizing this system must have a basic GPU for the pose estimation system to properly run.

3.2.2.2. Python Libraries and Environments

Required libraries, like PyTorch, are specified in a python notebook located in our GitHub repository. The environment for the system is a Google Colab interactive development environment.

3.2.3. Input Output Characteristics

3.2.3.1. Website GUI Input & Output

The input will show as a file sharing window that lets users upload videos (.mp4) and images (.png, .jpg, .jpeg) from their machine.

The output will be a playable video (.mp4) with the results of the fall detection system on display.

3.2.3.2. Training and Testing

Datasets containing videos of falls and normal activities are used to train the system. Training will make use of the UR Fall Data Set by the University of Rzeszow and the “Fall Detection dataset” made by Bournemouth University.

UR Fall contains 30 sets containing a sequence of images (representing a frame of video). Each image will be annotated with labels describing the current state of the person performing the action, such as “is falling” for a falling state. Bournemouth University’s dataset has a combined total of 21499 images.

Testing datasets were recorded by our team at the martial arts facility at Texas A&M. A total of 78 videos were recorded, each several seconds long with a quality of 480x640 15 FPS.

3.2.3.3. Video Output

The output shall display if a fall has occurred in a video by overlaying big red text stating “fall” on the image/video.

3.2.3.4. Failure Propagation

Any errors will be detected in the code and displayed to the user. Input videos will remain unaltered within the original database. A whole system crash could be resolved by rerunning the code.

Appendix A: Acronyms and Abbreviations

CNN	Convolution Neural Network
FPS	Frames per Second
GUI	Graphic User Interface

Appendix B: Definition of Terms

GitHub	Website for storing and managing repositories.
Bounding Boxes	Rectangle that surrounds a person or object.
Keypoints	Components of the human body, such as joints.

FALL DETECTION SYSTEM

Justin Haryanto

Nhan Nguyen

INTERFACE CONTROL DOCUMENT

REVISION – 3.0

30 April 2023

Interface Control Document

for

Fall Detection

Prepared by:

____

Author Date

Approved by:

____

Project Leader Date

____

John Lusher II, P.E. Date

____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	03 Oct 2022	Justin Haryanto		Draft Release
2.0	04 Dec 2022	Nhan Nguyen		Revision
3.0	30 Apr 2023	Justin Haryanto		Revision

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

No table of figures entries found.

1. Overview

This document describes the interfaces between the subsystems of the fall detection system. There are two interface sections for the GUI and fall detection subsystems. Videos are inputted into the GUI system. The results of this are inputted into the fall detection system. The ICD will explain in detail how the subsystems interface together to achieve the goal of the whole system.

2. References and Definitions

2.1. References

Pytorch - 1.12.1

TensorFlow - 2.10.0

AlphaPose - Sep 2022

2.2. Definitions

CNN	Convolution Neural Network
FPS	Frames per Second
GUI	Graphic User Interface

3. GUI

This subsystem acts as a convenient interface between the user and the fall detection subsystem by allowing video and image uploads into the system.

3.1. Website

A website acts as the GUI for the system. Users would upload images or videos to be sent to the fall detection system, then the fall detection system analyzes the input and returns results in the form of a video back to the website. The website possesses error checking to prevent problematic inputs, such as videos and images being uploaded simultaneously. It can also adjust its size to match whatever device screen it is displayed on.

4. Fall Detection

This subsystem uses information on key points and bounding boxes generated from the pose estimation subsystem processing the inputted video to determine if a fall has occurred.

4.1. Video Processing

Training videos have been divided into frames with labels indicating the action of the person in the video. However, it was found that videos that are too long caused AlphaPose to stall, therefore, videos will need to be kept short.

4.2. AlphaPose

AlphaPose has video processing built into its program and only needs to output to the fall detection system with the video frames being normalized to fit the detection system. This system is a multi-person keypoint detection system that can take video or camera inputs. It places bounding boxes on individuals and estimates the pose by feeding the bounding boxes to a pipeline.

AlphaPose was chosen because it can overcome several problems common in other systems, such as inaccurate bounding boxes. Its code is also convenient to work with as it has an implementation in google colab, a useful tool for collaboration.

4.3. Convolution Neural Network (CNN)

CNN is a deep learning algorithm that can recognize patterns in images. This makes it suitable for analyzing images and bounding boxes to determine if a fall has occurred. TensorFlow is used because it already has the framework set up for the CNN model.

4.4. Random Forest

Random forest is a simple classification model that can analyze key points and determine if a fall has occurred. Having an algorithm that is simple will reduce resource costs.

	1/25	2/01	2/08	2/15	2/22	3/01	3/08	3/22	3/29	4/05	4/12	4/19	4/26	4/29		
Status Update 1	🟩															
Status Update 2			🟩												0	Not Started
Status Update 3					🟩										1	In Progress
Status Update 4							🟩								2	Complete
Status Update 5									🟩						3	Behind Schedule
Final Design Presentation											🟩	🟩			4	Optional
Final Project Demo													🟩			
Final Report														🟩		
Pose Estimation and Fall Detection Integration	🟩	🟩	🟩	🟩												
Datasets for Training	🟩	🟩	🟩	🟩												
Training RF and CNN for better accuracy	🟩	🟩	🟩	🟩												
Refining Fall Detection Models			🟩	🟩												
Video Testing Sets			🟩	🟩	🟩	🟩										
Full System Integration				🟩	🟩	🟩										
Adding Datasets for Testing Full System				🟩	🟩	🟩										
System Testing							🟩	🟩	🟩							
Further Testing, Fixing, and Validation							🟩	🟩	🟩	🟩	🟩	🟩				
Make UI					🟩	🟩	🟩	🟩								
Refine UI								🟩	🟩	🟩	🟩	🟩				
Extra																
Live Video Input						🟪	🟪									
Live Video and Pose Estimation Integration								🟪	🟪							

	Success Criteria	Methodology	Responsibility
Data Sets	Process at least 15 UR and 5 Adhikari fall detection data sets	Having a variety of data sets would increase model accuracy	Justin Haryanto
Testing Sets	Create at least 10 480p videos with a duration of 10 seconds at 15 FPS compatible with the system	Testing the accuracy and processing time of the system when given videos at different frame rates	Justin Haryanto
User-Interface/Website	User able to submit an MP4 or collection of images to the UI, the UI should return the processed files as a MP4 and display to the user	Design UI with video submission box and a mechanism for outputting video and images	Justin Haryanto
Website Scaling	Website display on a variety of devices with appropriate layout adjustments	Test website on phone, tablets, and other devices	Justin Haryanto
Live Video Input	The system should accept continous series of video clips from a webcam at a minimum of 10 fps at 480p	Use cameras to provide live video feed to the model at various fps	Justin Haryanto
Fall Detection Accuracy	>90% of falls successfully detected	Use processed videos on the system, compare results to a specified labelled dataset	Nhan Nguyen
Fall Detection CNN Accuracy	>90% of fall images and bounding boxes successfully detected	Use fall images and bounding boxes on the CNN system, compared results to a specified labelled dataset	Nhan Nguyen
Fall Detection Random Forest Accuracy	>90% of fall keypoints successfully detected	Use key points on the RF system, compared results to a specified labelled dataset	Nhan Nguyen
Fall Detection Computation Time	Make Fall/Not Fall Decision in 1-30 seconds	Run individual images through fall detection system from labelled dataset	Nhan Nguyen
Fall Detection Computation Resources	Fall detection system take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	Run system on videos and images and record resource usage. Average file sizes up to 16 MB	Nhan Nguyen
Fall Detection Training Computation Time	Training Times is less than 24 hours	Record the time it take to prepare data and train machine learning models	Nhan Nguyen
Fall Detection Training Computation Resources	Training take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	Record system resouce usage during training.	Nhan Nguyen
Whole System Stress Test	Successfully process a stream of videos and images without system breaking down	Run system for 15 minutes with a variety of images and videos and check for any crashes or errors	Full Team
Whole System Functionality	Can the user submit files to the website and get back a display of the video and the fall detection results	Testing if the entire system works using the website UI to interface with the pose estimation and fall detection systems.	Full Team

FALL DETECTION SYSTEM

Justin Haryanto

Nhan Nguyen

SUBSYSTEM REPORT

REVISION – Draft

30 April 2023

Subsystem Reports

for

Fall Detection System

Prepared by:

Author Date

Approved by:

Project Leader Date

John Lusher II, P.E. Date

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	04 Dec 22	Nhan Nguyen		Draft Release
2.0	30 Apr 23	Justin Haryanto		Revision

Table of Contents

List of Tables

List of Figures

1. Introduction

The fall detection system is designed to detect when a person has fallen in a video or sequence of images. There are two main subsystems: the GUI, and the fall detection subsystems, which have both been evaluated on their performance.

This system works by allowing a user to upload a video/image from their machine through the GUI website to the fall detection system. The system feeds the video into the pose estimation system. During this part, the system extracts relevant key points and bounding box information, then sends this information to the fall detection machine learning models to determine if a fall has occurred. The results are then returned to the website.

These systems have been integrated into a whole system seen in the ConOps, ICD, and FSR, though they are not working to the best of their capability.

2. References and Definitions

2.1. References

Team 56 Fall Detection System (GitHub)

https://github.com/Nhannguyen993/TAMU-GitHub

2.2. Definitions

CNN	Convolutional Neural Network

3. Fall Detection Subsystem Report

3.1. Subsystem Introduction

A video or image sequence is given to the fall detection system via the website GUI. The pose estimation component of the system uses AlphaPose to extract key points and bounding box information from individual video frames for use in the fall detection machine learning (ML) component. The fall detection random forest and CNN models then analyze the frames and output results to be compiled into a video for the GUI to show.

The whole fall detection subsystem is currently being run on Google Colab and uses the GPU resource there, with the code being located on GitHub. The system has been evaluated on its ability to work on videos and images of the training data set collected from other universities. This is done to verify that key points and bounding box information could be extracted and analyzed for falls.

3.2. Subsystem Details

The pose estimation component of the fall detection subsystem can process videos with the use of AlphaPose, which is capable of whole-body pose estimation and tracking. It was found that AlphaPose performed well when run on COCO, a renowned data set for large-scale object detection, segmentation, and captioning, achieving a mean average precision of 75. The COCO dataset has hundreds of thousands of regular scenes containing common objects placed in natural context to allow programs to detect and segment these objects.

For determining falls, the random forest and convolutional neural network (CNN) machine learning models were used, where they were chosen based on the data to be given. Random forest is a simple model that is sufficient for the analysis of key point information. In contrast, the CNN model’s image processing capability made it suitable for the analysis of video image frames alongside the bounding box information.

Training and testing were performed using the UR Fall Detection dataset, which contains a series of falling videos with their respective labeled image frames. After the pose estimation subsystem generates the bounding box and key points information, the data is processed by the two machine learning models for analysis. Then the results of the models are compared, where a fall has occurred only if both models agree, and the verdict is outputted to an external system. For our demonstration, the external system is represented by a video overlaid with text indicating if a person has fallen or not.

3.3. Subsystem Validation

The following is the validation criteria for the subsystem:

Test	**Success Criteria**	**Methodology**
Fall Detection Accuracy	>90% of falls successfully detected	Use processed videos on the system, compare results to a specified labeled dataset
Fall Detection CNN Accuracy	>90% of fall images and bounding boxes successfully detected	Use fall images and bounding boxes on the CNN system, compared results to a specified labeled dataset
Fall Detection Random Forest Accuracy	>90% of fall keypoints successfully detected	Use key points on the RF system, compared results to a specified labeled dataset
Fall Detection Computation Time	Make Fall/Not Fall Decision in 1-30 seconds	Run individual images through fall detection system from labeled dataset
Fall Detection Computation Resources	Fall detection system take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	Run the system on videos and images and record resource usage. Average file sizes up to 16 MB
Fall Detection Training Computation Time	Training Times is less than 24 hours	Record the time it take to prepare data and train machine learning models
Fall Detection Training Computation Resources	Training take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	Record system resource usage during training.

Table 1: Fall Detection Validation Criteria

The fall detection system should be able to accurately detect a fall based on a video frame with data from the pose estimation system. The capability of the machine learning (ML) models in detecting a fall from the training/testing data is demonstrated for each data set used. Figure 1 and table 1 shows the results from a single frame and the accuracies of the ML models over the data sets from the University of Rzeszow (UR) and Bournemouth University (Adhikari). Figure 2 shows a successful fall detection on a frame from a video that does not resemble the ones used for training. Table 2 shows the accuracy of the whole subsystem when tested on newly created data, separate from previous training and testing data.

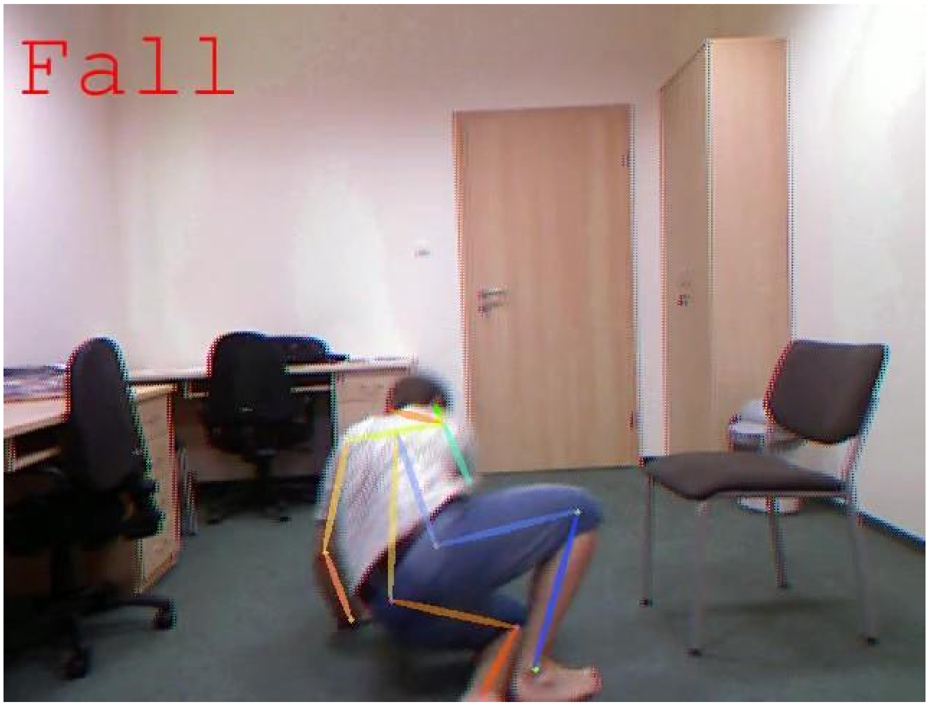


Figure 1: Fall Detected from Training/Testing Dataset

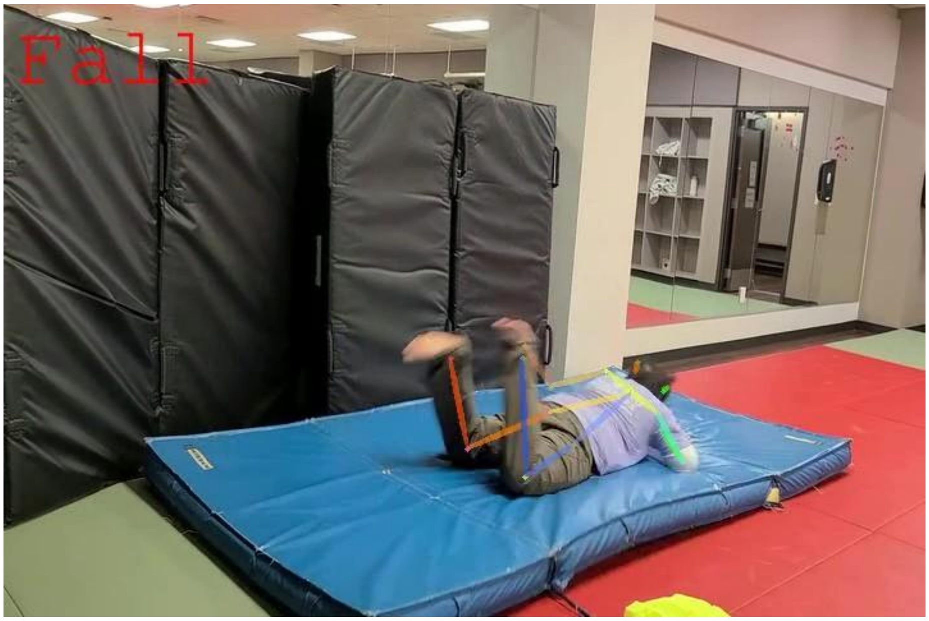


Figure 2: Fall Detected on Data Unlike Training/Testing Set

	UR Dataset	Adhikari Dataset	Both Datasets
RF	0.96	0.92	0.92
CNN	0.67	0.17	0.35

Table 2: Accuracies of ML models Over Testing Datasets

	Accuracy
Whole Subsystem	0.80

Table 3: Accuracies of ML models Over Newly Made Unseen Data

Test	Result
Fall Detection Computation Time	25 seconds
Fall Detection Computation Resources	25% GPU RAM,66% CPU RAM, and 33% Memory for avg video or series of images
Fall Detection Training Computation Time	Training Time is approx 1 hour for running over all training data sets once Training Time approx 7.5 minute for the average dataset
Fall Detection Training Computation Resources	95% GPU RAM,51% CPU RAM, and 18% Memory for training on part of data set at any one time

Table 4: Computation Times and Resources

3.4. Subsystem Diagnostic

The fall detection system has faced rigorous training and testing with mixed results. Both the ML models were able to achieve relatively high accuracies on the UR data set, with RF doing well on the Adhikari dataset while CNN suffers a significant drop in accuracy. When tested on newly created falling videos that the models have not seen during training, the general accuracy of the fall detection system is 80 percent. However, the subsystem has not been thoroughly tested to examine performance on motions similar to falls or edge cases like a person catching themselves before falling and so could potentially output false positives/negatives. The computation times and resources are matching expectations with the exception of the training computation resources.

3.5. Subsystem Mitigation

An expanded data set with videos with a greater variety of falls/fall-like motions and further refinement of the ML model parameters could serve to improve accuracy. Training for edge cases for fall-like movements could serve to prevent false positives/negatives. Reevaluating how training is done could reduce the computation resources needed.

3.6. Subsystem Conclusion

The subsystem's ability to detect falls with reasonable accuracy is working correctly, though the CNN's accuracy is still poor and could be improved upon. Its ability to handle videos showing edge cases for falling movements have not been thoroughly tested. When interfaced with the rest of the subsystems it will be able to analyze inputted videos and return results on falls.

4. Externals Subsystem

4.1. Subsystem Introduction

Externals covers the website GUI and dataset collection components of the project.

Website GUI allows the user to upload files and get back results from the fall detection system. Dataset collection involves collecting, creating, and processing datasets for use in training and testing the system.

4.2. Subsystem Details

A website is used as the GUI of the system so that it can communicate with the fall detection system directly. Hosting is done using Ngrok, while the creation of the website was made using a combination of Flask, HTML, and CSS.The website also has file extension error checking (to ensure that the user submits the correct video and image files) and UI scaling (to adjust the website to fit the shape of the screen to maintain presentability).

The design is based on a typical Texas A&M website and made to be as simple as possible while being attractive. A&M's brand guide was referred to for fonts, colors, and regulations, while A&M's Electrical Engineering website was referenced for its design and logo.

Datasets for the training of the machine learning models were collected from authors who previously researched fall detection, mainly researchers from the University of Rzeszow and Bournemouth University. Both sources have several dozen sets containing over a hundred images, each labeled with the type of motion being performed by the person in the set. (such as standing or falling).

Testing datasets were made by the team, featuring recorded videos of a person falling at the Texas A&M Martial Arts facility. This dataset was used to evaluate the performance of the fall detection system on inputs it has never seen before.

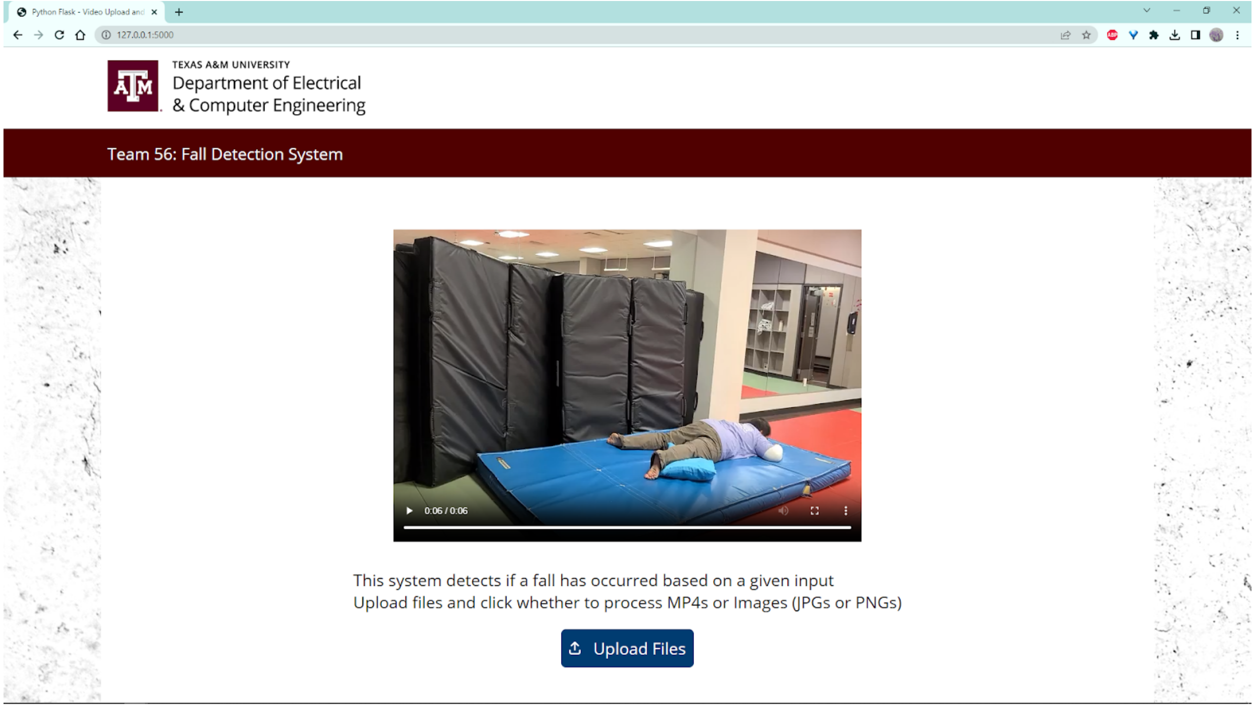
4.3. Subsystem Validation

The following are the validation criteria for the Externals subsystem:

Test	**Success Criteria**	**Methodology**
Website GUI	User able to submit an MP4 or collection of images to the UI, the UI should return the processed files as a MP4 and display to the user	Design UI with video submission box and a mechanism for outputting video and images
UI Scaling	UI should scale depending on the screen size the user is interacting with the system from	Makes the system more useable on a wider variety of devices
Training Datasets	Process at least 15 UR and 5 Adhikari fall detection data sets	Having a variety of data sets would increase model accuracy
Testing Datasets	Create at least 10 480p videos with a duration of 10 seconds at 15 FPS compatible with the system	Testing the accuracy and processing time of the system when given videos at different frame rates

Table 5: Externals Validation Criteria

Validation tests for the website are mainly to ensure functionality, which it met in the figure below.



This system detects if a fall has occurred based on a given input
Upload files and click whether to process MP4s or Images (JPGs or PNGs)

Figure 3: Website GUI with Output Video (prior to system integration)

For testing the datasets component, there should be enough datasets to properly train the fall detection system and enough testing videos to ensure the machine learning models get a new test each time. By the end of the project, a total of 78 datasets were collected from academic sources, while 89 video clips were recorded. Both were also processed into a usable format for the system to be trained and tested on.

4.4. Subsystem Diagnostic

Our website was capable of accepting user uploads and checking that they were the correct file types, displaying the output of the system, and scaling to fit the screen size. However, only one user can use it at a time and there were network issues outside our control.

The datasets that were used only had one person and displayed them in a controlled environment, thus its ability to work for other types of situations was not tested.

4.5. Subsystem Mitigation

A more reliable method of communication needs to be found and error checking for network issues is to be added to ensure stable operation. The ability for multiple users to use the system should also be implemented.

There should also be more variety in the datasets for both training and testing, especially since the datasets used feature only one person in a controlled environment. Future datasets should include multiple people, varying lighting conditions, different video qualities, and new camera angles.

4.6. Subsystem Conclusion

The Externals subsystem met the conditions it sought out for. For the website GUI component, that was to provide an easy and attractive way to interact with the fall detection system. For the datasets component, it was to provide

FALL DETECTION SYSTEM

Justin Haryanto

Nhan Nguyen

SYSTEM REPORT

REVISION – 1.0

30 April 2023

System Reports

for

Fall Detection System

Prepared by:

____

Author Date

Approved by:

____

Project Leader Date

____

John Lusher II, P.E. Date

____

T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
1.0	30 Apr 23	Nhan Nguyen		Draft Release

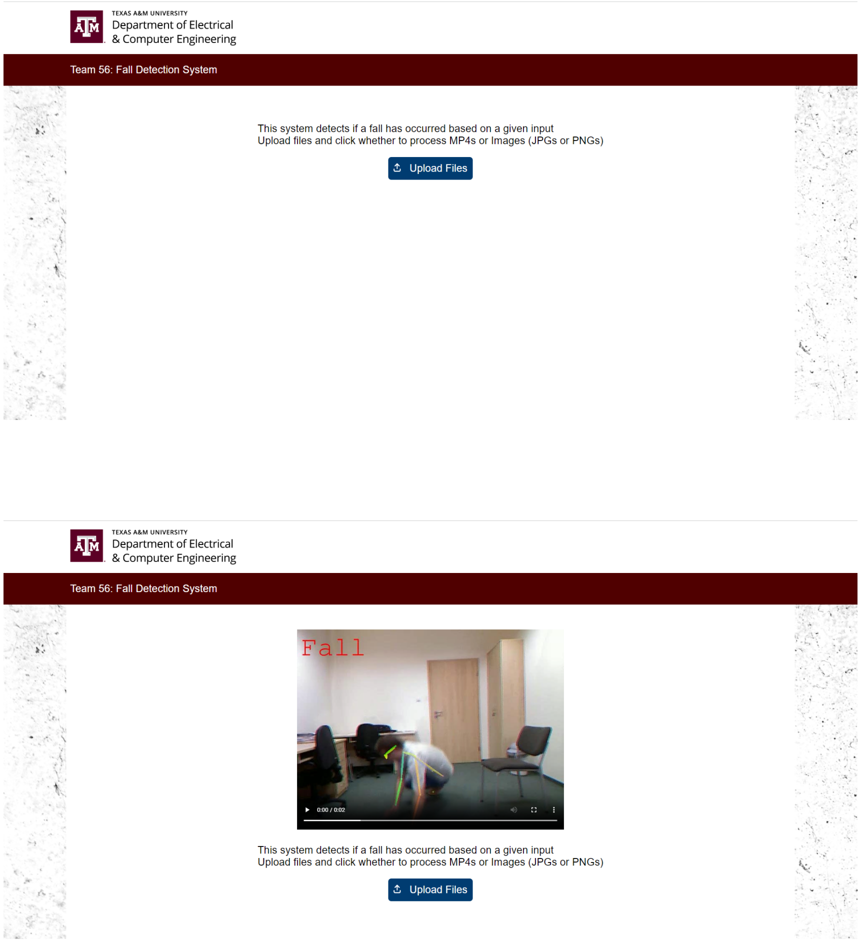
Table of Contents

List of Tables

List of Figures

1. Overview

The sponsor requested a video based fall detection system which utilized pose estimation. This meant a system where a user could provide video input and receive results on whether the people inside the video have fallen or not with good accuracy. The final result of this is seen below in figure 1. Here the user could upload a video or series of images into the website and receive results in the form of a playable video where the uploaded file is edited to include the word "Fall" to indicate when a fall occurred. The results page also allows users to upload more files for further tests and the website in general could adjust its layout to meet the size of the screen it is displayed on.



TEXAS A&M UNIVERSITY
Department of Electrical
& Computer Engineering

Team 56: Fall Detection System

This system detects if a fall has occurred based on a given input
Upload files and click whether to process MP4s or Images (JPGs or PNGs)

TEXAS A&M UNIVERSITY
Department of Electrical
& Computer Engineering

Team 56: Fall Detection System

This system detects if a fall has occurred based on a given input
Upload files and click whether to process MP4s or Images (JPGs or PNGs)

Figure 1: Website Home Page and Results Page

The system was created as two main subsystems, the Fall Detection subsystem and the Externals subsystem. A website GUI from the external subsystem allows users to upload videos and images to the Fall Detection system. The Fall Detection system processes an inputted video/image through a pose estimation system called AlphaPose to extract results on keypoints and bounding boxes for each frame. This data is sent to two machine learning (ML) models, random forest and CNN, to be analyzed for any human performing falling motions. These models were trained from collected data sets, a part of the externals. The results from this analysis are sent back to the website GUI as a video for the user to review. This system report is to allow for the review of decisions and benchmarks of the development of the fall detection system, particularly the integration of the subsystems.

2. Development Plan and Execution

2.1. Design Plan

The initial design was just the video processing and pose estimation system connected to a fall detection as seen below in figure 2. This did not originally include any form of user interface. We initially plan on refining the pose estimation system, AlphaPose, to be better able to handle unusual poses made by humans, and having the video processing as a more separate component. There are other features such as live video input.

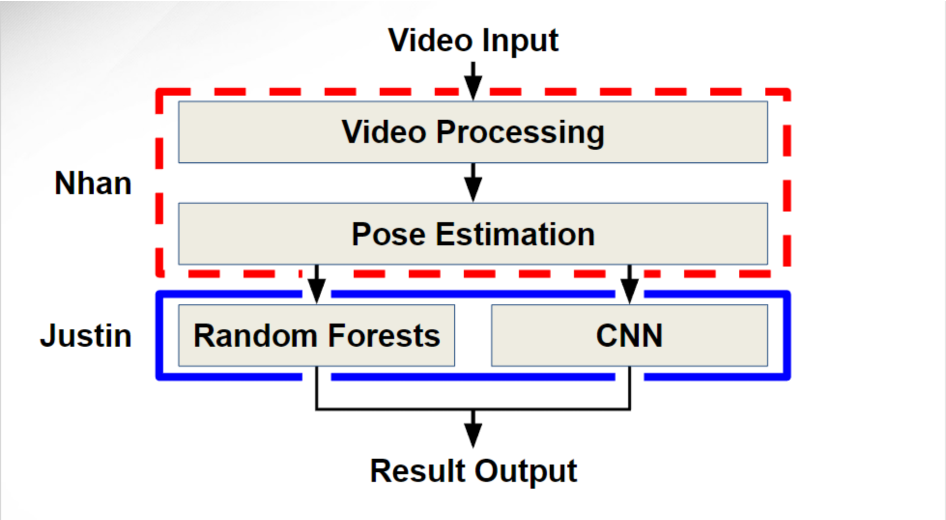


Figure 2: Original Design

Improving AlphaPose proved beyond our ability and the need for a GUI became more clear once we started integrating the systems and encountering issues with interacting with the fall detection code implemented on Google Colab. This led to the restructuring of the system design seen below in figure 3, with a Website UI as the GUI. Features like live video input were determined to be optional.

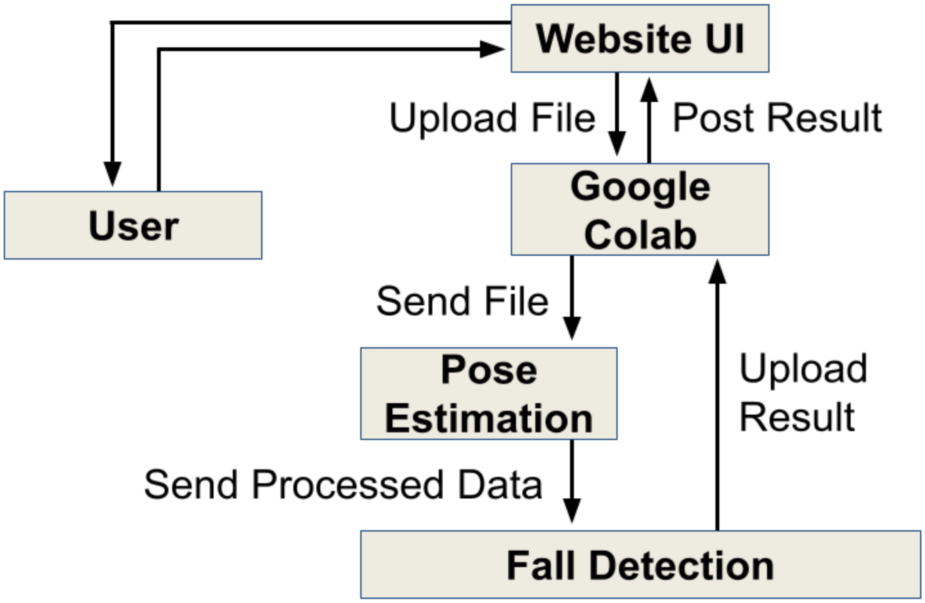
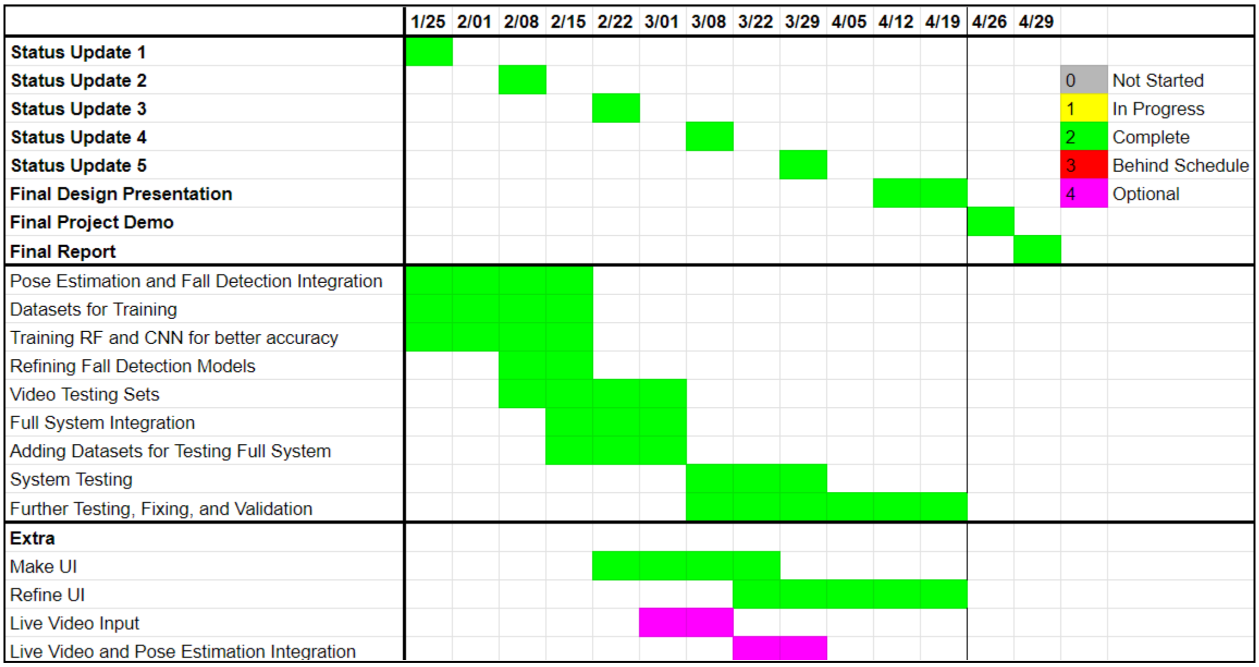


Figure 3: New Design

2.2. Execution

Before integrating, we started by finalizing the subsystems. This involves planning out, implementing, and troubleshooting the new website GUI. The fall detection subsystem was further refined with the pose estimation and machine learning models better integrated than last year, and refining the models to be more accurate by altering their parameters and retraining.

Integration begins once the subsystems are usable. The Pose Estimation component is properly integrated with the Fall Detection system, making the full Fall Detection System. Datasets were further collected and used to train and refine the ML models of this system. Video testing sets were collected to evaluate the whole system. The website GUI is integrated with the full Fall Detection system to make up the whole system. Testing is done, with further refinements, troubleshooting, and validation.



	1/25	2/01	2/08	2/15	2/22	3/01	3/08	3/22	3/29	4/05	4/12	4/19	4/26	4/29		
Status Update 1	🟩															
Status Update 2			🟩												0	Not Started
Status Update 3					🟩										1	In Progress
Status Update 4							🟩								2	Complete
Status Update 5									🟩						3	Behind Schedule
Final Design Presentation											🟩	🟩			4	Optional
Final Project Demo													🟩			
Final Report														🟩		
Pose Estimation and Fall Detection Integration	🟩	🟩	🟩	🟩												
Datasets for Training	🟩	🟩	🟩	🟩												
Training RF and CNN for better accuracy	🟩	🟩	🟩	🟩												
Refining Fall Detection Models			🟩	🟩												
Video Testing Sets			🟩	🟩	🟩	🟩										
Full System Integration				🟩	🟩	🟩										
Adding Datasets for Testing Full System				🟩	🟩	🟩										
System Testing							🟩	🟩	🟩							
Further Testing, Fixing, and Validation							🟩	🟩	🟩	🟩	🟩	🟩				
Extra																
Make UI					🟩	🟩	🟩	🟩								
Refine UI								🟩	🟩	🟩	🟩	🟩				
Live Video Input						🟪	🟪									
Live Video and Pose Estimation Integration								🟪	🟪							

Figure 4: Execution Plan

2.3. *Validation Plan*

The full validation plan is shown below. The fall detection system was tested for accuracy and computation time/resources. The accuracy of the random forest, CNN, and whole fall detection system was partly validated. The computation time/resources were partly validated. The number of collected data sets and testing sets were validated. The website user interface and functions like website scaling were validated. Whole system testing and functionality was partly validated.

	Specifications	**Results**
Data Sets	Process at least 15 UR and 5 Adhikari fall detection data sets	30 UR and 10 Adhikari fall datasets
Testing Sets	Create at least 10 480p videos with a duration of 10 seconds at 15 FPS compatible with the system	78 MP4 clips with a duration of 5 to 7 seconds
User-Interface/ Website	User able to submit an MP4 or collection of images to the UI, the UI should return the processed files as a MP4 and display to the user	Designed a website, based on a typical A&M website, that allowed for mp4/images upload and result return
Website Scaling	Website display on a variety of devices with appropriate layout adjustments	Website could autoscale based on device it is displayed on
Live Video Input	The system should accept continuous series of video clips from a webcam at a minimum of 10 fps at 480p	None, specification is optional
Fall Detection Accuracy	>90% of falls successfully detected	80% accuracy on own videos
Fall Detection CNN Accuracy	>90% of fall images and bounding boxes successfully detected	67% accuracy on UR dataset,17% accuracy on Adhikari dataset, 35% accuracy on both Adhikari and UR Fall Datasets
Fall Detection Random Forest Accuracy	>90% of fall keypoints successfully detected	96% accuracy on UR dataset, 92% accuracy on Adhikari dataset, 92% accuracy on both Adhikari and UR Fall Datasets
Fall Detection Computation Time	Make Fall/Not Fall Decision in 1-30 seconds	25 seconds

Fall Detection Computation Resources	Fall detection system take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	25% GPU RAM,66% CPU RAM, and 33% Memory for avg video or series of images
Fall Detection Training Computation Time	Training Times is less than 24 hours	Training Time is approx 1 hour for running over all training data sets once, Training Time approx 7.5 minute for the average dataset
Fall Detection Training Computation Resources	Training take up 75% or less of system GPU RAM, CPU RAM,and Memory on average	95% GPU RAM,51% CPU RAM, and 18% Memory for training on part of data set at any one time
Whole System Stress Test	Successfully process a stream of videos and images without system breaking down	Website have trouble displaying videos, possibly due to network issues
Whole System Functionality	Can the user submit files to the website and get back a display of the video and the fall detection results	Successfully submitted video/image files and receive results

Table 1: Validation Plan

3. Conclusion

The final design is a functional fall detector capable of determining if a fall has taken place in a video frame with reasonable accuracy. Many problems and challenges were encountered during the development of this system. The original design and plans needed to be rehauled into something more usable and within the team’s capabilities. The new design allows for greater flexibility as the team could more easily use the Website GUI to work with the fall detection system situated in Google Colab. The resulting product is easier to use and meets most of the validation specifications, though there is room for improvement.

4. Future Work

Improvements to the machine learning models accuracy could be done, particularly for the CNN model. The models could also be further improved by training on a more diverse dataset which includes various motions that resemble but are not falls, or with unusual poses. This would reduce the potential false positives or negatives in the system. The system is also fairly slow, taking seconds to minutes to process videos, so research in speeding up the system could be examined. Furthermore the website GUI occasionally experiences network issues when displaying results, a problem that could be rectified by further development.