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

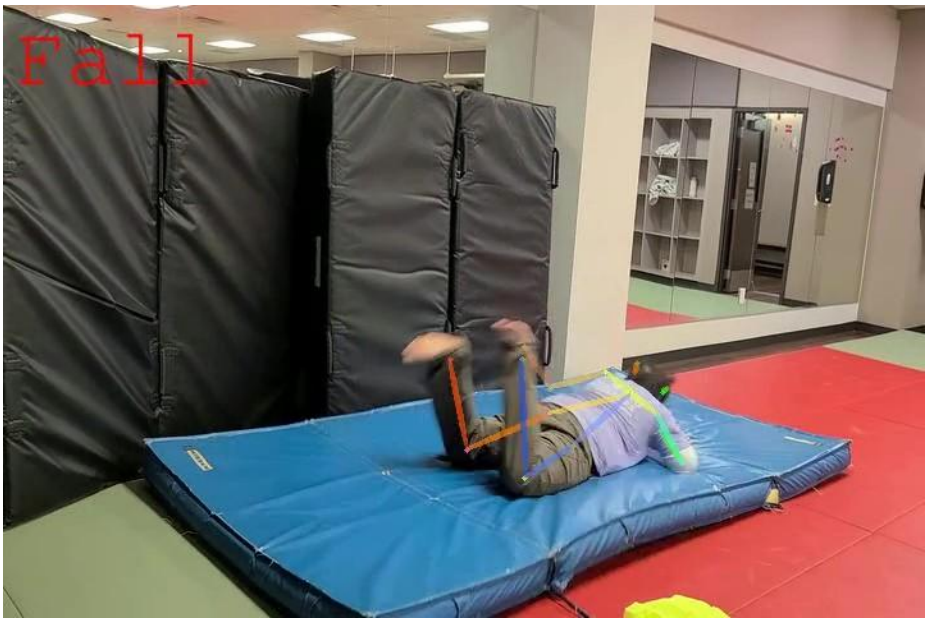

Figure 2: Fall Detected on Data Unlike Training/Testing Set

	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.

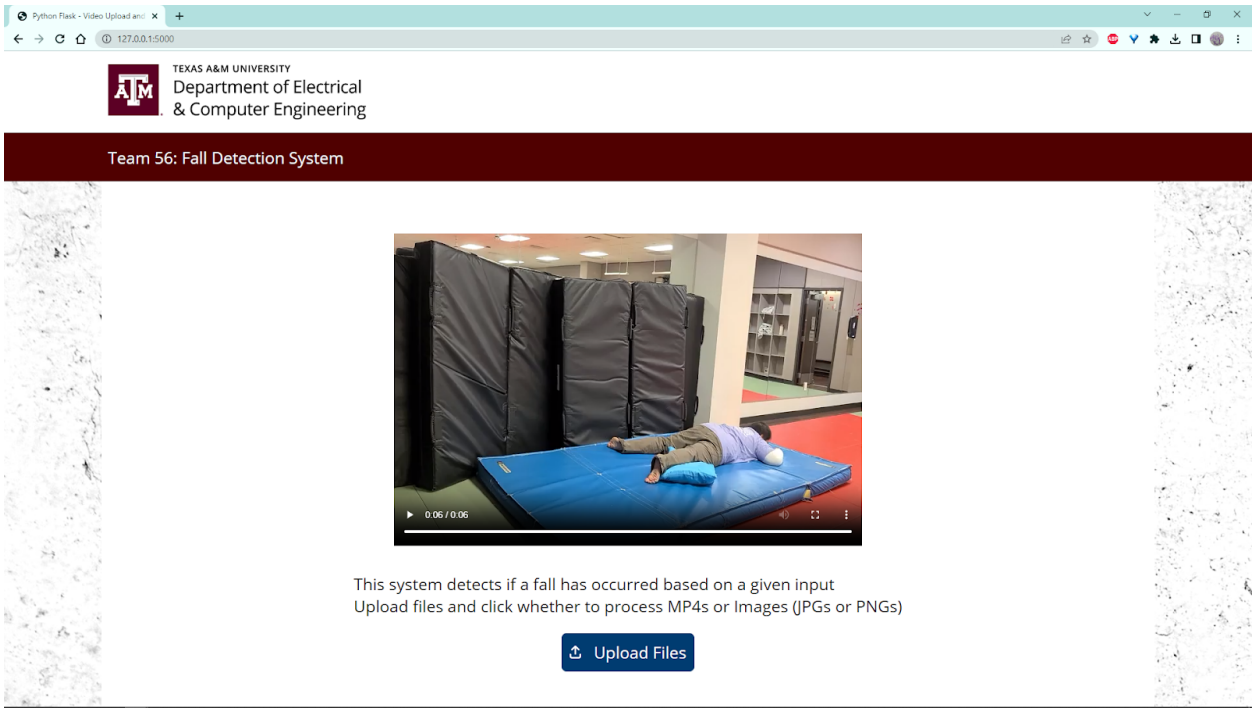

Figure 3: Website GUI with Output Video (prior to system integration)

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