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.

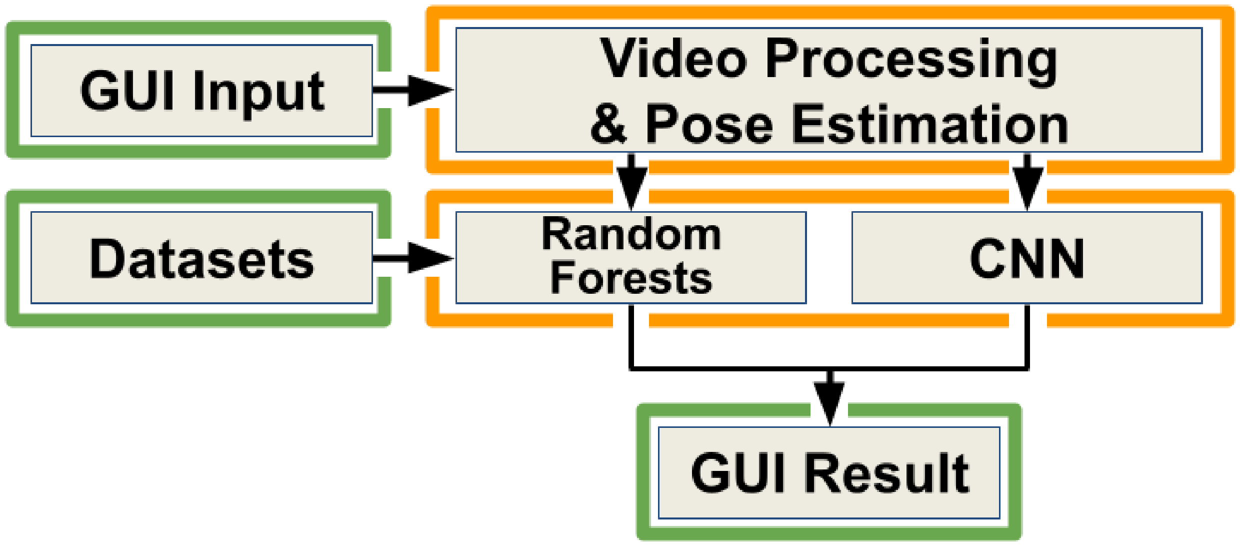

Figure 1: Distribution of Responsibilities

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.

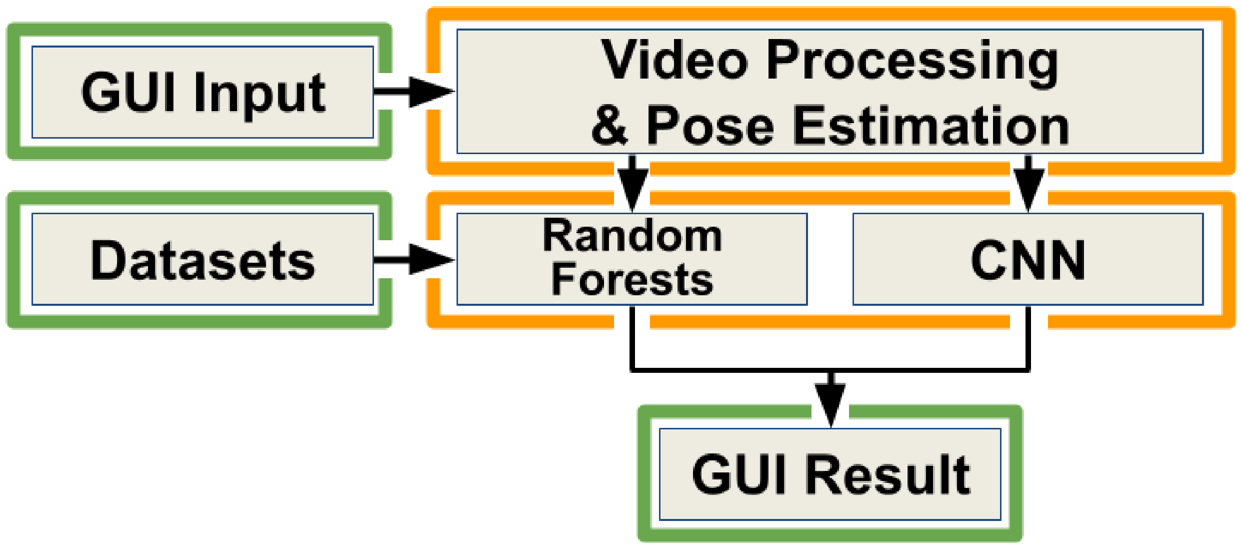

Figure 2: Block Diagram of System

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.