Fall Detection

Justin Haryanto
Nhan Nguyen

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – Draft
03 October 2022

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
0	03 Oct 2022	Nhan Nguyen		Draft Release
1	04 Dec 2022	Justin Haryanto		Revision

Table of Contents

List of Tables

No table of figures entries found.

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 function with low quality videos.

The system shall utilize pose estimation to identify people in the video, where it will assign keypoints and bounding boxes to them. The fall detection system will then detect falls and output if a fall has occurred in the video. The results should be at least comparable to pre-existing fall detection systems. Figure 1 displays the integration of the project in the proposed ConOps.

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
- Justin Haryanto: Fall Detection

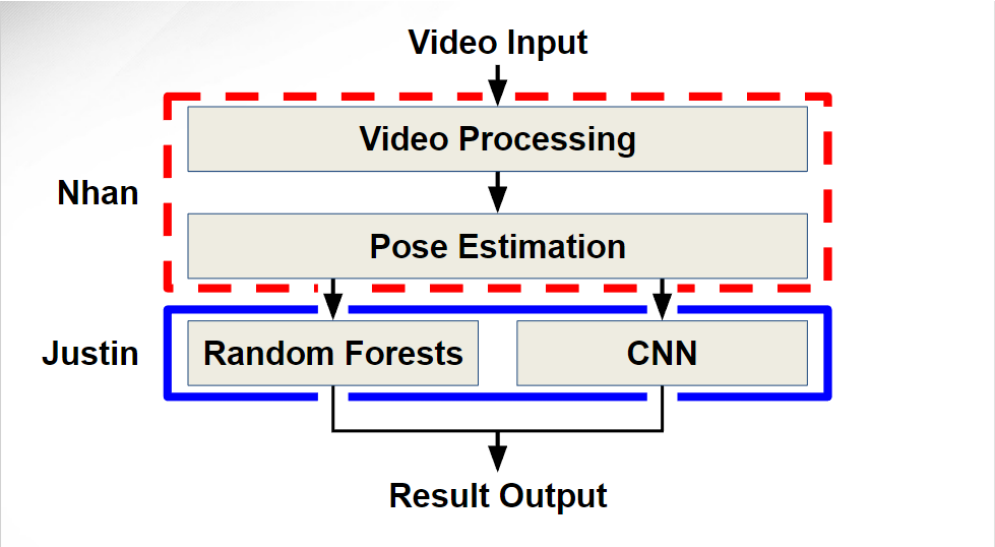

Figure 1: Distribution of Responsibilities

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:

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	12/04/2022	https://github.com/Nhannguyen993/TAMU-GitHub

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. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

3.1. System Definition

The fall detection system comes in two main subsystems: pose estimation and fall detection. The pose estimation subsystem converts 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. The fall detection subsystem determines if a fall has occurred and sends that information to an external program.

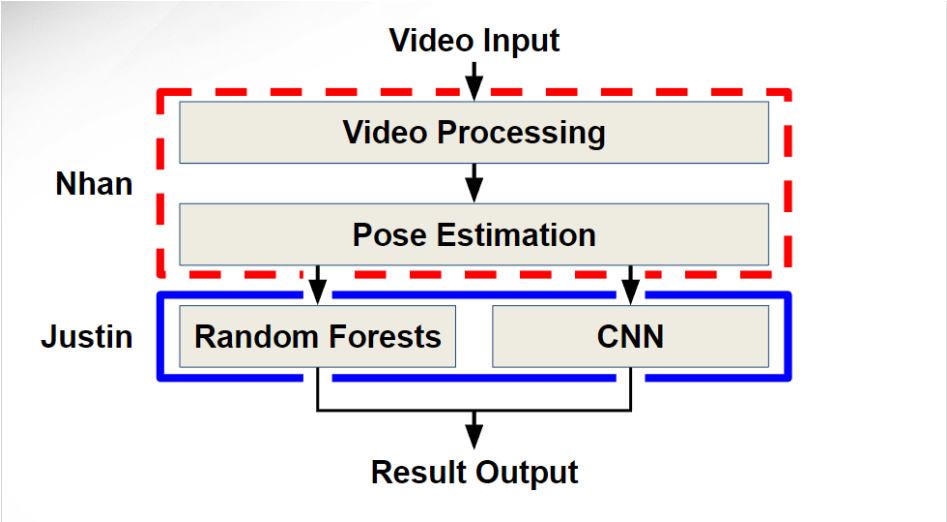

Figure 2: Block Diagram of System

3.1.1. Video Processing

AlphaPose already performs video processing by itself.

3.1.2. Pose Estimation

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.

3.1.3. 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. Pose Estimation Video Processing Requirement

The pose estimation system can process recorded videos and live feeds at a max of 30 frames per second (FPS). It can apply bounding boxes and keypoints accurately regardless of video quality.

Rationale: 30 FPS is regularly used for TV, live broadcasting, and online services. Higher frame rates are unnecessary for the most part and takes up more data, which slows down the program.

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 Jupyter notebook interactive development environment.

3.2.3. Input Output Characteristics

3.2.3.1. Training and Testing Input

Datasets containing videos of falls and normal activities are used to train the system.

One dataset that will be used is the UR Fall Data Set by the University of Rzeszow, which contains 30 fall sequences of video and labeled frames. Each frame of the video will be annotated with labels such as “is falling” describing the current state of the person performing the falling action.

Another dataset is the “Fall Detection dataset” made by Bournemouth University, which have 21499 images. The input for testing will be similar with the addition of videos with falls in more natural situations to test the robustness of the system.

3.2.3.2. Pose Estimation Video Processing

The pose estimation subsystem shall process a video with a maximum frame rate of 30 FPS.

3.2.3.3. Video Output

The output shall display if a fall has occurred in a 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

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.