Système de détection des chutes en ligne

par

Oussema KESKES

MÉMOIRE PAR ARTICLES PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M. Sc. A.

MONTRÉAL, LE 14 DÉCEMBRE 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE:

Mme Rita Noumeir, Directeur de mémoire
Département de génie électrique, École de technologie supérieure

M. Stéphane Coulombe, Président du jury
Département de génie logiciel et des TI, École de technologie supérieure

Mme Catherine Laporte, Examinateur externe
Département de génie électrique, École de technologie supérieure

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 25 NOVEMBRE 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

En premier lieu, je remercie ma directrice de recherche, professeure Rita Noumeir, de m'avoir accueillie dans son équipe de recherche et pour sa patience, sa disponibilité et surtout ses conseils précieux.

Un grand merci à mes parents et mes soeurs, pour leur amour, et leur soutien inconditionnel, à la fois moral et économique, qui m'a permis de réaliser les études que je voulais et par conséquent ce mémoire.

Je remercie également mes collègues du laboratoire LATIS notamment Haytham, ThanhDung, Jihad, George, Nicolas et Alban pour leurs conseils.

Système de détection des chutes en ligne

Oussema KESKES

RÉSUMÉ

L'automutilation et le suicide dans les centres correctifs canadiens présentent un problème sérieux. Ce type d'incidents a augmenté durant ces dernières années, d'où la nécessité d'avoir de nouvelles techniques de prévention et de détection pour assurer l'intervention immédiate et réduire les impacts de suicides. Les méthodes les plus connues de suicide sont la surdose, la pendaison et la coupe du poignet. À travers cette recherche, on vise de trouver une technique appropriée de détection de chute qui suit une tentative de suicide par surdose.

Notre but est de développer une méthode qui détecte la chute en temps réel d'une manière efficace et fiable avec un taux minimal de fausses alarmes. La plupart des méthodes de détection proposées dans la littérature utilisent des techniques traditionnelles qui limitent la fiabilité et la généralité du système. En outre, ces méthodes ne fonctionnent pas en temps réel, elles n'acceptent que des vidéos bien segmentées et qui ne contiennent qu'une seule action.

Pour atteindre notre objectif, nous avons utilisé la Kinect v2 de Microsoft qui est une caméra RGB-D. Ce capteur propose différents flux de données : image RGB, infrarouge, nuage de points et les données squelettiques humaines. Nous avons choisi d'utiliser les données du squelette humain pour détecter la chute puisqu'elles sont indépendantes de l'environnement et des conditions d'éclairage.

Quatre bases de données publiques ont été utilisées pour évaluer notre système. Ces bases de données sont : TSTv2 et FallFree qui sont des bases de données conçues pour la détection de chutes ; NTU-RGB+D et PKU-MMD qui sont des bases de données orientées vers la détection et la reconnaissance d'actions.

La méthode utilisée dans la présente recherche se compose de deux modules : un détecteur d'action et un détecteur de chute. Le premier module est responsable de détecter le début et la fin de chaque action. Ce module est basé sur l'algorithme d'apprentissage profond 1D Convolution Neural Network (1D-CNN). Il découpe le flux continu des données squelettiques en sous-séquences de quinze images et classifie chaque sous-séquence indépendamment. Si l'algorithme détecte un mouvement dans la sous-séquence, cette dernière est classifiée comme positive. La succession de sous-séquences classifiées comme positives forme une action. Ce module a été entrainé avec la base de données PKU-MMD. Après l'extraction de l'action, le rôle du deuxième module est de la classifier en 'chute' ou 'non chute'. Ce module est basé sur l'algorithme Spatial Temporal Graph Convolution Networks (ST-GCN). Il a l'avantage de tenir compte simultanément les informations spatiales et temporelles. Il exploite le fait que les squelettes forment des graphes au lieu de grilles en 2D ou 3D. Premièrement il a été entrainé avec la base NTU-RGB+D. Puis, la technique de transfert d'apprentissage a été appliquée en utilisant les deux bases de données TSTv2 et FallFree. Les performances obtenus dépassent celles des méthodes proposées dans la littérature sur les deux bases de données TSTv2 et FallFree.

Mots-clés: détection de chutes, temps réel, ST-GCN, 1D-CNN, transfert d’apprentissage, apprentissage profond, TSTv2, FallFree, NTU-RGB+D, PKU-MMD, Kinect v2.

Real-time fall detection system

Oussema KESKES

ABSTRACT

Self-injury and suicide in Canadian correctional facilities is a serious problem. This type of incident has increased in recent years and requires prevention and detection techniques to provide immediate intervention and reduce its impact. The most well known methods of suicide are overdose, hanging and cutting the wrist.This work focuses on the detection of the falls following an overdose suicide attempt.

Our goal is to develop a method that detects the fall in real time in an efficient and reliable manner with a minimal false alarm rate. Most of the detection methods proposed in the literature use traditional techniques that limit the reliability and generality of the system. Additionally, these methods don't work in real time, they only accept well segmented videos that contain only one action.

To achieve our goal, we used the Kinect v2 from Microsoft which is an RGB-D camera. This sensor offers different data streams : RGB image, infrared, point cloud and human skeletal data. We chose to use the human skeleton to detect the fall since it is independent of the environment and lighting conditions.

Four public databases were used to train our system : the TSTv2 and FallFree which are fall detection oriented databases ; plus the NTU-RGB+D and PKU-MMD which are action detection and recognition oriented databases.

Our method is composed of two modules : an action detector and a fall detector. The first module is responsible for detecting the beginning and the end of each action. This module is based on the deep learning algorithm 1D-CNN. It cuts the continuous flow of skeletal data into subsequences of 15 images and classifies each subsequence independently. If the algorithm detects motion in the subsequence, the subsequence is classified as positive. The succession of positively classified subsequences forms an action. This module was trained with the PKU-MMD database. After extraction of the action, the role of the second module is to classify it as 'falling' or 'not falling'. This module is based on a ST-GCN. It has the advantage of simultaneously capturing spatial and temporal information. It exploits the fact that the skeletons are in the form of graphs instead of 2D or 3D grids. First it was trained with the NTU-RGB+D base. Then, the transfer learning technique was applied using both TSTv2 and FallFree databases. The performance obtained exceeds that of the methods proposed in the literature on both TSTv2 and FallFree datasets.

Keywords: Fall detection, real-time, ST-GCN, 1D-CNN, transfer learning, Deep learning, TSTv2, FallFree, NTU-RGB+D, PKU-MMD, Kinect v2.

TABLE DES MATIÈRES

Page

INTRODUCTION 1

CHAPITRE 1 REVUE DE LITTÉRATURE 5
1.1 Axe 1 : Système de détection de chute hors ligne 5
1.2 Axe 2 : Systèmes de détection de chute en temps réel 7

CHAPITRE 2 VISION-BASED FALL DETECTION USING ST-GCN 11
2.1 Abstract 11
2.2 Introduction 12
2.2.1 Literature review 16
2.3 Methods 20
2.3.1 Architecture of the network 21
2.3.2 Dataset 22
2.3.3 Training 25
2.4 Results 26
2.4.1 Training and Evaluation Protocol 27
2.4.2 Evaluation metrics 28
2.4.3 Results 30
2.4.4 Comparison with state-of-the-art 31
2.4.4.1 Comparison with results using the TST v2 dataset 31
2.4.4.2 Comparison with results using the FallFree dataset 34
2.4.4.3 Comparison with results using the NTU-RGB+D dataset 34
2.5 Discussion 36
2.6 Conclusion and future works 38

CHAPITRE 3 REAL-TIME SKELETON BASED FALL DETECTION SYSTEM 41
3.1 Abstract 41
3.2 Introduction 42
3.3 Methods 46
3.3.1 Action detection 47
3.3.1.1 Network architecture 47
3.3.2 Fall detection 48
3.4 Dataset 49
3.4.1 Dataset and labelling 49
3.4.2 Evaluation 50
3.4.2.1 Evaluation metrics 50
3.4.2.2 Training and evaluation protocol 50
3.5 Experiments 51
3.5.1 Action detection 51
3.5.2 Online fall detection system 51

3.6 Results 52
  3.6.1 Action detection 52
  3.6.2 Online fall detection system 55
3.7 Discussion 55
3.8 Conclusions 56

CONCLUSION ET RECOMMANDATIONS 59

ANNEXE I DOCUMENTATION AND REPRODUCIBILITY OF CODES 63

BIBLIOGRAPHIE 68

LISTE DES TABLEAUX

Page

Table 2.1 RGB-D fall datasets 22

Table 2.2 Training and evaluation protocol 29

Table 2.3 Results of our fall detection method 31

Table 2.4 Comparison between our method and other state-of-the-art methods tested on the TST v2 dataset 35

Table 2.5 Comparison between our method and other state-of-the-art methods tested on the Fallfree dataset 36

Table 2.6 Comparison between our method and other state-of-the-art methods tested on the NTU RGB-D dataset 37

Table 3.1 Performance of the action detection module on PKU-MMD dataset. We report the results before and after applying the interpolation step 54

Table 3.2 Results of the online fall detection system 55

LISTE DES FIGURES

Page

Figure 0.1 Evolution des incidents de surdose dans les établissements fédéraux au Canada 2

Figure 2.1 Skeleton joints provided by Kinect v2 15

Figure 2.2 Taxonomy of fall detection system 17

Figure 2.3 Spatial temporal graph convolution network architecture 22

Figure 2.4 The spatial-temporal graph of skeleton sequence ; inter-body edges (blue) and intra-body edges (black) 23

Figure 2.5 Transfer learning process 26

Figure 3.1 Real-time fall detection system 47

Figure 3.2 Action detection module architecture 48

Figure 3.3 Labelling of PKU-MMD dataset 50

Figure 3.4 Action detection module classification errors 53

Figure 3.5 Confusion matrix before applying an interpolation step 54

Figure 3.6 Confusion matrix after applying an interpolation step 54

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

1D-CNN	1D Convolution Neural Network
3D-CNN	3D Convolution Neural Network
ADL	Activity of daily living
CNN	Convolutional Neural Network
COM	Center Of Mass
CUSUM	CUmulative SUM
EDM	Euclidean Distance Matrix
ELS	Error in Sast Sub-sequence of the action
EMA	Error in classifying a sub-sequence in the Middle of the Action
FMCW	Frequency-Modulated Continuous-wave radio
FPS	Frames Per Second
GA	Genetic Algorithm
GA	Gaussian mixturemode
IR	Infrared
KNN	K-nearest neighbors
LSTM	Long Short-Term Memory
NCA	Neighborhood Component Analysis
RNA	Report Nonexistent Action
RNN	Recurent Neural Network
SGD	Stochastic Gradient Descent
ST-GCN	Spatial Temporal Graph Convolution Networks
SVMA	Sum Vector Magnitude of acceleration
SVM	Support Vector Machine

SWT	Stationary Wavelet Transform
RF	Radio Frequency
RF	Random Forest
ToF	Time-of-Flight
WT	Wavelet Transform

LISTE DES SYMBOLES ET UNITÉS DE MESURE

FN	False Negatives
FP	False Positives
TN	True Negatives
TP	True Positives

INTRODUCTION

Le présent travail fait partie d'un grand projet intitulé “An intelligent vision-based system for detecting self-harm behavior ” élaboré en collaboration avec Aerosystems International dans le cadre d'une compétition organisée par le gouvernement canadien. Ce projet vise à offrir un outil de détection d'automutilation dans les prisons canadiennes basé sur des algorithmes d'intelligence artificielle. Dans ce travail, nous nous concentrerons sur la détection de la chute qui suit la perte de conscience causée par une surdose.

Le suicide présente un problème sérieux dans la vie notamment pour les personnes sous surveillance. La plupart des études ont montré que plus d'une personne détenue sur cinq avait tenté de se suicider (Kouyoumdjian, Schuler, Matheson & Hwang, 2016). Bien que les stratégies et les méthodes qui ont été proposées et mises en œuvre dans le but de limiter l'accès aux méthodes de suicides, on observe toujours une augmentation du nombre de tentatives de suicide. Les comportements d'automutilation qui se produisent en prison consomment des ressources médicales et provoquent des perturbations considérables (Smith, Kaminski, Power & Slade, 2019).

Le moyen le plus répandu de se suicider dans les prisons est la surdose. Comme indiqué dans (Kouyoumdjian *et al.*, 2016), le nombre d'incidents de surdose a augmenté de 40 en 2012-2013 à 88 en 2016-2017 dans les établissements fédéraux au Canada. La figure 0.1 présente l'évolution d'incidents de suicides entre les années 2012 et 2017. La perte de conscience est un signe courant d'une surdose. Elle survient lorsqu'une personne n'est plus capable de rester consciente et devient incapable de répondre aux stimuli. Par conséquent, elle tombe instantanément et semble dormir. Une intervention d'urgence et immédiate est nécessaire quand une personne perd conscience. Une méthode fiable de détection de chute est donc nécessaire pour assurer une intervention instantanée et réduire les risques de blessure.

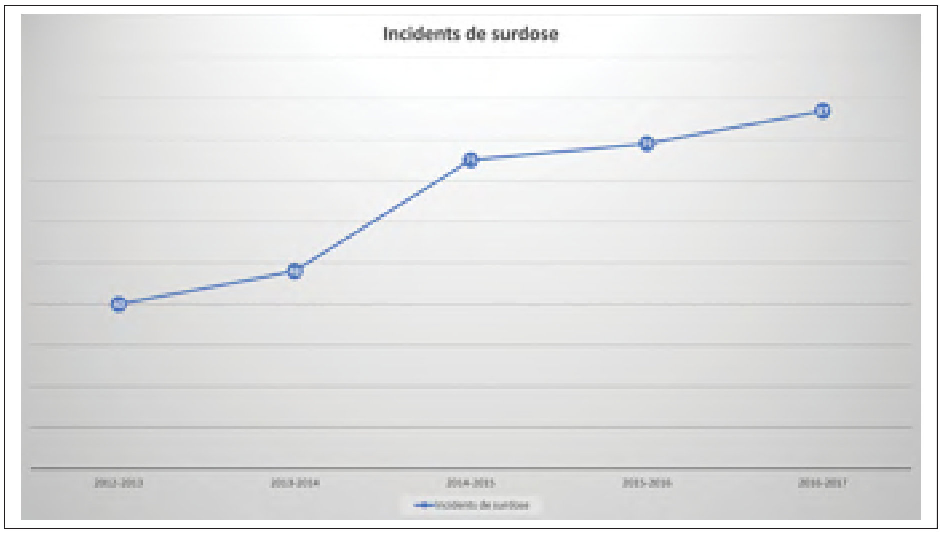


Figure 0.1 Evolution des incidents de surdose dans les établissements fédéraux au Canada

Beaucoup de travaux ont proposé des systèmes de détection de chute en utilisant différents capteurs et différents algorithmes. Dans la plupart de ces travaux, les méthodes sont développées pour fonctionner hors ligne, ç'est-à-dire, ils ne peuvent pas accepter un flux de données continues comme des vidéos en continu. Par ailleurs, Une grande partie d'eux utilisent des algorithmes basiques et traditionnels qui ont été entrainés et testés sur une seule base de données, ce qui limite leurs efficacité et généralité. À cet effet, il est necessaire d'avoir un nouveau système intelligent qui détecte la chute en temps réel et qui soit à la fois efficace et robuste.

Dans cette thèse, nous proposons un système de détection de chute en temps réel basé sur deux algorithmes d'intelligence artificielle : 3D Convolution Neural Network (1D-CNN) et Spatial Temporal Graph Convolution Networks (ST-GCN). Les entrées de notre modèle sont des coordonnées de squelettes provenant d'une caméra RGB-D Kinect v2. Ce type de données a l'avantage d'être indépendant de l'environnement et des conditions d'éclairage. Notre système est composé de deux modules :

- module de détection d'action : ce module est responsable de détecter et d'extraire les actions à partir d'un flux de vidéo continu et de les passer au module de détection de chute pour détecter la présence d'une chute. Ce module est basé sur le réseau de neurones 1D-CNN. Nous avons utilisé la base de données PKU-MMD pour entrainer et tester ce module ;

- module de détection de chute : ce module est responsable de classifier des séquences de vidéo en deux classes : chute et non-chute. Ce module est basé sur réseau de neurones ST-GCN. Ce modèle a prouvé sa performance dans le domaine de la reconnaissance d'actions (Kong, Li, Zhang, Ni & Han, 2019b). Les bases de données de détection de chute disponibles sont de petite taille comparées aux bases de données de reconnaissance d'actions. D'autre part, une grande quantité de données était nécessaire pour entrainer un algorithme d'apprentissage profond. De ce fait, nous avons décidé d'utiliser la technique de transfert d'apprentissage du domaine de la reconnaissance d'actions au domaine de détection de chute.

Afin de garantir la robustesse et la généralité de notre système, nous allons utiliser quatre bases de données :

- NTU-RGB+D : une grande base de données adaptée au domaine de reconnaissance d'action. Elle contient 56 000 échantillons appartenant à 60 différentes classes ;
- TSTv2 : une base de données adaptée au domaine de détection de chute. Elle contient 264 échantillons appartenant à 8 classes ;
- Fallfree : une base de données adaptée au domaine de détection de chute. Elle contient 391 échantillons appartenant à 12 classes ;
- PKU-MMD : une grande base de données adaptée au domaine de détection et de reconnaissance d'action. Elle contient 1076 vidéos, et chaque vidéo contient environ 20 différentes actions appartenant à 51 classes.

Voici les principales contributions qui ont été apportées dans le cadre ce projet de recherche :

- Nous proposons un système de détection de chute en temps réel qui offre de très bons résultats.

- nous mettons en œuvre un algorithme d'apprentissage profond pour détecter le début et la fin de l'action en temps réel.
- nous appliquons la technique d'apprentissage par transfert et une combinaison de différentes bases de données pour pallier le problème de la taille limitée des bases de données disponibles ;
- nous proposons un module de détection de chute qui peut être utilisé sur de nouvelles données sans avoir besoin de refaire le processus d'entrainement tout en conservant une bonne précision ;

CHAPITRE 1

REVUE DE LITTÉRATURE

L'apparition des caméras de profondeur a stimulé les recherches vers la détection de chute basée sur la vision par ordinateur. Après sa sortie en juillet 2014, Kinect v2 est devenu le capteur le plus utilisé dans les systèmes de détection de chutes (Xu, Zhou & Zhu, 2018). Cette caméra est composée d'un capteur RGB, d'un capteur infrarouge (IR) basée sur la technologie ToF (time-of-flight) et d'un microphone. Kinect V2 fournit différents flux de données : image RGB, image de profondeur, image IR, données squelette et audio. Notre revue de littérature est basée sur deux axes. Le premier axe se concentre sur les travaux qui ont proposé des systèmes de détection de chute traditionnels qui fonctionnent hors ligne. Le deuxième axe se concentre sur les travaux de détection de chute en temps réel.

1.1 Axe 1 : Système de détection de chute hors ligne

La plupart des recherches qui ont été réalisées dans le cadre de détection de chute sont des méthodes qui fonctionnent en mode hors-ligne. Ces méthodes ont été entrainées et testées à l'aide de séquences bien segmentées où chaque séquence contient une seule action.

La plupart des systèmes de vision par ordinateur sont basés sur les squelettes et les articulations du corps humain pour détecter les activités quotidiennes humaines en raison de leur robustesse au changement d'éclairage et à la variation de scénario (Yan, Xiong & Lin, 2018).

(Gasparrini, Cippitelli, Gambi, Spinsante, Wåhslén, Orhan & Lindh, 2015) ont choisi d'utiliser à la fois les données squelettiques de Kinect v2 et les données d'accélération d'un accéléromètre monté sur la taille du sujet. Ils ont exploité la variation de la position de l'articulation du squelette, la distance de la base de la colonne vertébrale par rapport au sol et l'amplitude de l'accélération mesurée par l'accéléromètre. Les auteurs ont opté pour une méthode basée sur des seuils. Ils ont utilisé la base de données TST v2 pour évaluer leur approche et ils ont rapporté un taux d'exactitude de 99%. Les résultats obtenus sont très bons, mais cette méthode présente deux inconvénients. Le premièr est que la méthode du seuil n'est pas robuste, car les seuils eux-mêmes dépendent des caractéristiques du sujet telles que le sexe et l'âge (Yao, Yang & Huang, 2019). Le deuxième est dans le choix des capteurs. L'utilisation de la caméra et de l'accéléromètre à la fois est complexe et nécessite une synchronisation. De plus, l'accéléromètre doit être installé en accordant une attention particulière à son orientation.

(Tsai & Hsu, 2019) ont proposé une méthode qui combine features-based method et les machines à vecteurs de support. Ils ont utilisé cinq caractéristiques calculées à partir de quatre articulations du squelette (tête, cou, centre de l'épine et base de la colonne vertébrale). L'avantage de cette méthode est que le seuil est appris automatiquement par l'algorithme. Ils ont rapporté un taux d'exactitude de 93,56% sur la base de données TST v2. Bien que le seuil soit calculé automatiquement, le taux d'exactitude reste limitée par les choix de caractéristiques par rapport aux méthodes d'apprentissage profond qui apprennent automatiquement les caractéristiques les plus pertinentes.

(Mastorakis, 2018) ont proposé et implémenté un système de détection de chute sur la plate-forme NVIDIA Jetson Tx2. Ilont décidé d'extraire les importantes articulations du squelette de l'image de profondeur au lieu d'utiliser directement le joint fourni par Kinect puisque la carte Jetson TX2 ne prend pas en charge le SDK de la Kinect. Ensuite, ils ont utilisé un 1D-CNN pour classifier l'action. Ils ont atteint un taux d'exactitude de 99,2% sur la base de données NTU-RGBD. Ils ont utilisé tous les échantillons de chute et ils ont sélectionné au hasard des échantillons d'autres classes pour construire la classe non-chute. Bien que ce jeu de données soit bien connu dans le domaine de la reconnaissance d'activité, il n'est pas recommandé de l'utiliser dans le problème de détection des chutes. Il présente certaines limites, par exemple il ne couvre pas tous les types de chutes (Le, Morel et al., 2014).

(Le *et al.*, 2014) ont utilisé les cordonnées des données squelettiques et le plan du sol de la chambre fournis par Kinect pour calculer la distance et la vitesse de la tête et du colonne vertébrale par rapport au sol. Puis, ils ont utilisé le modèle SVM pour la classification. Ils ont obtenu 0% de faux positifs et 3,3% de faux négatifs, ce qui est un résultat très intéressant.

L'ensemble de données utilisées dans ce travail n'est pas public, nous ne pouvons donc pas vérifier les résultats et l'auteur a mentionné qu'il avait supprimé des échantillons qui ressemblent aux chutes (comme les actions assises et allongées sur le sol) pour obtenir de meilleurs résultats.

La plupart des travaux précédents utilisaient des fonctionnalités qui sont extraites manuellement puisque ça ne nécessite pas une grande base de données et permet de se concentrer explicitement sur les fonctionnalités qui contribuent largement à la tâche spécifique. Cependant, il peut être difficile de proposer des caractéristiques aussi raisonnables et cohérentes pour certaines tâches ; cela nécessite une connaissance approfondie du domaine. D'autre part, nous avons maintenant accées aux méthodes d'apprentissage en profondeur qui ont surpassé les méthode de sélection de caractéristiques manuelles puisqu'elles peuvent extraire des fonctionnalités plus pertinentes durant l'entrainement ; mais ça nécessite beaucoup plus de données.

1.2 Axe 2 : Systèmes de détection de chute en temps réel

Dans la vie réelle, le modèle recevra un flux continu de données contenant différentes actions avec des instants de début et de fin inconnus. En premier lieu, le modèle doit détecter le début et la fin de l'action, puis il doit la classer. Le modèle doit effectuer ces deux tâches en temps réel. Contrairement aux systèmes de détection de chute conventionnels qui observent la séquence entière avant de fournir une classification de l'action, une détection de chute en temps réel doit détecter la chute à la volée à partir d'un flux de données continu. La détection de chute en temps réel devient un nouveau défi. Dans certaines recherches, les auteurs affirment que leur modèle est suffisamment rapide pour fonctionner en temps réel. Quelques travaux sont implémentés en temps réel.

(Tsai & Hsu, 2019), ont proposé une méthode de détection des chutes basée sur la caméra Kinect v2. Ils ont extrait les coordonnées du squelette à partir de l'image de profondeur. Un modèle CNN a été utilisé pour classifier l'action d'entrée. Ils ont atteint un taux d'exactitude de 99,2%. Les auteurs ont mentionné que leur méthode est assez rapide et a été mise en œuvre en temps réel, mais ils n'ont mentionné ni la procédure de l'implémentation ni les nouveaux résultats.Les performances peuvent se dégrader lorsque la méthode est appliquée en temps réel.

(Chenguang, Hu, Min, Deng, Zou & Min, 2020) ont proposé une méthode de détection des chutes en temps réel en utilisant une caméra monoculaire. Ils ont utilisé modèle de mélange gaussien (GMM) pour détecter la personne. Ensuite, deux ellipses ont été utilisées pour encadrer la tête et le torse. Puis, ils ont calculé trois caractéristiques pour chacune : l'angle d'inclinaison de l'ellipse, la vitesse et le rapport des axes court et long de l'éclipse. Enfin, ils ont utilisé le modèle CNN pour déterminer la corrélation entre les deux ellipses. Cette méthode a été testée sur leur propre base de données et atteint un taux d'exactitude de 90,5%. Ils ont mené une expérience en temps réel mais ils n'ont mentionné pas si leur base de données contient des vidéos segmentées ou continues.

(Wu, Wang, Cheng, Li, Chen & Zhou, 2019) ont encodé les articulations du squelette en images RGB. Ensuite, ils les alimentent à un modèle CNN. Pour évaluer leur méthode, ils ont utilisé une combinaison de la base de données MSRDaily Activity3D (Li, Zhang & Liu, 2010) et leur propre base de données acquis à l'aide de la caméra Kinect v2. Ils ont atteint un taux d'exactitude de 93,75% en mode hors ligne. Une méthode de vote a été utilisée pour détecter les chutes en temps réel. Chaque ensemble de 15 images est classé comme chute ou non-chute. Si trois des cinq dernières ensembles sont classées comme chute, une chute est détectée. Ils ont testé leur approche en utilisant 20 séquences et ils ont rapporté un taux d'exactitude de 95%. La principale limite de ce travail est que le jeu de données utilisé ne contient que des vidéos segmentées, c'est-à-dire que chaque vidéo ne contient qu'une seule action.

(Triantafyllou, Krinidis, Ioannidis, Metaxa, Ziazios & Tzovaras, 2016) ont proposé une détection de chute en temps réel dans un environnement de travail intérieur. Ils ont extrait trois caractéristiques à partir des images de profondeur : la vitesse verticale, la variance de la zone et la taille de la personne. La chute est détectée à l'aide du modèle de markov caché. L'ensemble de données a été acquis à l'aide de deux caméras Kinect v2 dans deux environnements de travail différents.

Les chutes sont détectées avec un taux d'exactitude de 97,14%. Cette méthode repose sur des fonctionnalités extraites manuellement qui limitent la robustesse et la généralité du modèle.

(Musci, De Martini, Blago, Facchinetti & Piastra, 2018) ont opté pour une méthode basée sur un accéléromètre pour détecter les chutes. Ils ont utilisé la base de données SisFall (Sucerquia, López & Vargas-Bonilla, 2017) pour entrainer un réseau de neurones récurrents avec mémoire à long terme (LSTM). Ils ont divisé toutes les séquences SisFall en fenêtres de 1,28 s avec un chevauchement partiel de 50% et ils ont étiqueté manuellement les données pour les adapter à leur modèle. Leur méthode a atteint un taux d'exactitude de 97,16%. Malgré les bons résultats, ce travail repose sur des accéléromètres qui doivent être portés toute la journée. Ceci est inconfortable et peut être altéré par le prisonnier.

CHAPITRE 2

VISION-BASED FALL DETECTION USING ST-GCN

Oussema Keskes[1] , Rita Noumeir[2]

[1] Département de génie logiciel et des technologies de l'information, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Département de Génie Électrique, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article soumis à la revue « IEEE Access » en octobre 2020.

2.1 Abstract

Falls are a growing issue in society and has become a hot topic in the healthcare domain. Falls are more likely to occur due to age or health problems such as cardiovascular issues and muscle weakness. In this work, we focus on fall detection. The after effects of falls often lead to the use of prescription pain medication. We are motivated to help prevent suicide attempts by overdose in the Canadian correctional services. Most previous studies were based on hand-crafted features which limit the robustness and generality of the system. We therefore propose a general vision-based system, using Spatial Temporal Graph Convolutional Networks (ST-GCN). This system has proven its effectiveness and robustness in the action recognition domain. Contrary to previous models, this model can be applied directly to new data without the need to retrain the model while offering good accuracy. Additionally, with the help of transfer learning, we can solve the insufficient data problem. By using three public datasets : the NTU RGB-D dataset, the TST Fall detection dataset v2 and the Fallfree dataset to validate our method, we achieved a 100% accuracy, surpassing the state-of-the-art.

keywords - Suicide prevent, Unconsciousness, Fall detection, Deep learning, ST-GCN, RGB-D data, Skeleton, TST Fall detection dataset v2, Fallfree, NTU RGB-D

2.2 Introduction

Suicide and self-harm behavior present a serious problem in real life and especially for people under surveillance. The most widespread route to suicide in prisons is narcotic overdose. As stated by (McKendy, Biro & Keown, 2019), the number of overdose incidents in federal custody in Canada increased from 40 in 2012/2013 to 88 in 2016/2017. Unconsciousness is a common sign of an overdose, occuring when a person is unable to maintain his/her awareness of their surroundings and becomes unable to respond to stimuli. Consequently, the person falls down instantly and appears to be asleep. Urgent intervention is needed immediately after a person loses consciousness. A reliable falling motion detection method is therefore needed to reduce the risk of injury and death.

Previous fall detection models generally used a threshold-based method to detect falls by comparing the input data with a setting reference value (Ren & Peng, 2019). A threshold is a limit above which a fall event is detected. For example, in methods using accelerometer sensors, features are calculated from three axial acceleration measurements. The sensor can be mounted on any part of the human body, such as the wrist or pelvis. The Sum Vector Magnitude of acceleration (SVMA), given by Equation (2.1), is widely used in the threshold-based method with accelerometer data (Delahoz & Labrador, 2014) :

$$SV(t) = \sqrt{a_x^2(t) + a_y^2(t) + a_z^2(t)} \tag{2.1}$$

where a_x, a_y and a_z represent the accelerometer measurements of three axes at instant t. The alarm is triggered when the calculated SVMA at instant t exceeds a fixed value. The value of the threshold is set by researchers based on their observations of several experiments. With a high threshold, some fall events will be missed, while a low threshold will increase the false alarm rate. In the latter case, the algorithm will be sensitive to any sudden change in body movement, such as picking up something from the floor, and will consider it as a fall event.

Despite the advantage of the low computational complexity of this method, the optimal choice of the threshold value presents a real challenge and significantly affects the performance of the system. The threshold depends on human body characteristics such as height, age and sex. A threshold can be appropriate for one person and inappropriate for another, which is why different threshold values are chosen in the literature even though they use the same configuration (Ren & Peng, 2019).

After 2014, the use of threshold-based methods in the literature decreased from 51% to 16%. This method became inadequate to reliably achieve the desired goal, especially with the appearance of new advanced sensors like cameras (Xu *et al.*, 2018).

In order to overcome the problem of selecting the optimal threshold, researchers began using machine learning algorithms such as Support Vector Machine (SVM), k-nearest neighbors (KNN). Classical machine learning algorithms became the new mainstream in fall detection systems (Xu *et al.*, 2018). A comparison was made between the threshold-based method and machine learning algorithms for fall detection in (de Quadros, Lazzaretti & Schneider, 2018). The data were collected from an accelerometer, a gyroscope and a magnetometer located on the subject's wrist. The authors conclude that machine learning provides much better results than threshold-based methods. Unlike the threshold-based method, machine learning algorithms explore the input data to guess the hypothesis that best fits the data. SVM is a well-known machine learning modèle. It is based on the idea of finding the optimal hyperplane that divides the input data into different classes, in our case "fall" and "no fall". For example, the authors in (Chen, Li, Wang, Hu & Ye, 2020), used the Mask R-CNN to detect and extract the subject from the RGB image, then they used a CNN for features extraction.

Although they achieved higher accuracy compared to the threshold-based method (Ren & Peng, 2019), machine learning methods are still based on hand-crafted features that are manually engineered by researchers. Hand-crafted features are based on choosing features directly related to a fall and could be used to distinguish falling from other actions (such as the velocity and acceleration of joints). The main drawback of this approach is that nothing guarantees that the chosen features are good descriptors for classification. In addition, it is not robust to classify an action with similar characteristics, as with jumping (Xu *et al.*, 2018). The choice of the features is not straightforward and requires a good knowledge of the domain under consideration. The challenge is the choice of features that best reflect the essence of a fall instead of just one aspect of a fall (Xu *et al.*, 2018).

With the development of deep learning, we no longer need to use hand-crafted features, as the algorithm will automatically explore data and extract useful features for the specific classification task. For instance, (Hwang, Ahn, Park & Park, 2017) developed a method based on a 3D-CNN. They used depth map data, collected using Kinect v2, to train their model. (Xu & Zhou, 2018) use skeleton data to calculate the accelerated velocity of the Center of Mass (COM) of different body parts and feed it to a Long Short-Term Memory network (LSTM) to detect falls.

However, deep learning algorithms require a large training dataset, and sometimes obtaining the necessary amount of data is costly and laborious.

We propose a vision-based fall detection system using only skeleton data. This data stream is obtained from the RGB-D camera of Microsoft's "Kinect v2". The Kinect v2 returns 25 joints numbered 0–24, as shown in figure 2.1. Each joint has 3 coordinates (x, y, z).

Contrary to RGB data, skeleton data has the advantage of being invariant to illumination and background variation. Additionally, a skeleton sequence provides trajectories of human motion using only a small number of joint positions. Therefore, it offers the advantage of low computational volume.

(Yan *et al.*, 2018) propose a new deep learning method called the Spatial-Temporal Graph Convolution Network (ST-GCN). As indicated by its name, it has the advantage of capturing spatial and temporal information simultaneously. It exploits the fact that the skeletons are in the form of graphs instead of 2D or 3D grids, and achieves great results in the domain of action recognition. By exploiting the fact that fall detection is related to the action recognition problem, we can benefit from using transfer learning to enhance the effectiveness of our method. In the first step, we train the ST-GCN algorithm on the NTU-RGB+D (Shahroudy, Liu, Ng & Wang, 2016) dataset to acquire the relevant features for action recognition. This is an RGB+D action

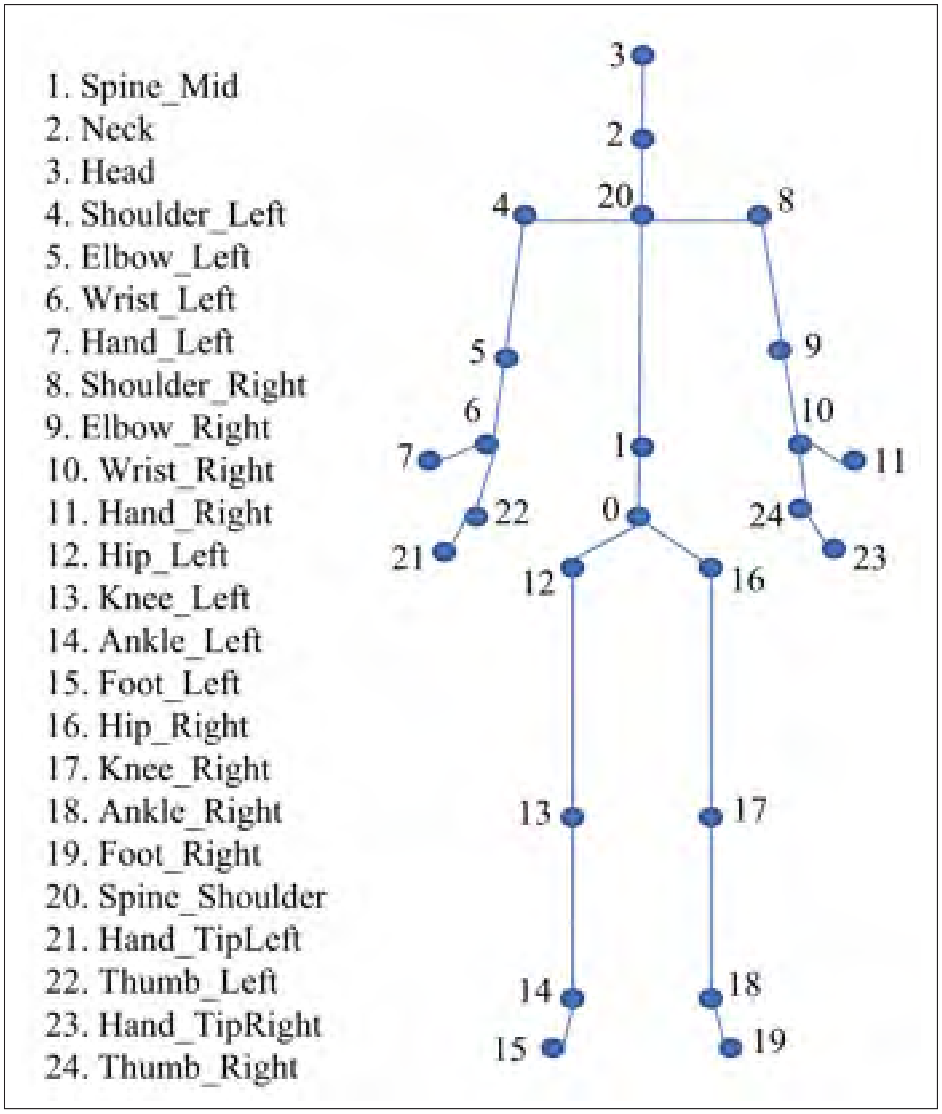


Figure 2.1 Skeleton joints provided by Kinect v2

recognition-oriented dataset that contains 60 different actions. In the second step, we apply transfer learning to the fall detection domain using two datasets : the TST Fall detection dataset v2 (Enea, Ennio, Samuele & Susanna, 2016) and the Fallfree dataset (Alzahrani, Jarraya, Salamah & Ben-Abdallah, 2017). These are two RGB+D fall detection oriented datasets which contain different types of falls and activities of daily living (ADL).

Through this paper, we will present the following main contributions :

- to the best of our knowledge, this paper is the first to apply the ST-GCN on Fall detection. Many techniques have been used to encode temporal information, such as using optical flow, 3D-CNN and RNN/LSTM and have obtained some promising results, but they still suffer from limitations and there is still no perfect method for temporal encoding. Optical flow techniques are computationally intensive, 3D filters in 3D-CNN algorithms have a very rigid temporal structure, and the weight-sharing mechanism of RNN/LSTM algorithms make sequence matching imprecise (Wang, Li, Ogunbona, Wan & Escalera, 2018). To move beyond such limitations, we use the Spatial-Temporal Graph Convolution Network (ST-GCN), which has proven its effectiveness to handle the spatial configuration of joints as well as their temporal dynamics. This method has achieved promising results in the action recognition domain (Yan *et al.*, 2018) ;
- we apply the transfer learning technique to address the limited size of the available datasets. Despite the efforts made to provide a high quality and a large dataset, the available fall dataset is not large enough to train a deep learning algorithm (Xu *et al.*, 2018). We aim to reduce the gap between the available dataset and the data-hungry deep learning algorithm by using transfer learning. This is done by transferring the learned knowledge from the action recognition domain to the fall detection domain ;
- retraining the model when facing new data is not practical and it is time consuming (Zhang, Li & Ogunbona, 2017). Therefore, we suggest a model that can reuse the previous knowledge for new data with no need to retrain the model, while offering good accuracy. To achieve this goal, the ST-GCN algorithm, the transfer learning technique and two different fall datasets were used together to provide a general fall detection system able to deal with new data.

2.2.1 Literature review

Xu et al. (Xu *et al.*, 2018) conducted a survey on fall detection technologies, proposing a taxonomy for fall detection methods based on the sensor used. Three major categories were proposed as shown in figure 2.2 : accelerometer-based, radio frequency (RF) sensor-based and vision-based.

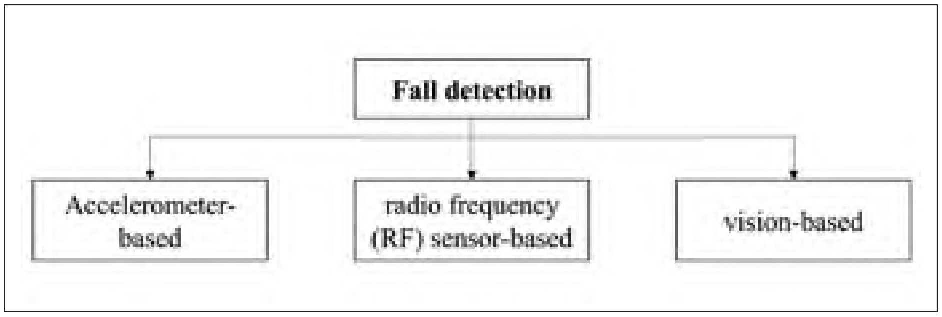


Figure 2.2 Taxonomy of fall detection system

The first category refers to accelerometer devices and accelerometers built into smartphones. (Wang, Zhang, Li, Lee & Sherratt, 2014) authors used an accelerometer sensor to detect a fall. They calculate the SVMA of the subject and compared it to a predefined threshold. If the SVMA value surpasses the threshold, they calculate the heart rate using a pulse pressure sensor and the trunk angle. The system will contact emergency services if these two values are above normal. They achieved an accuracy of 97.5%. Accelerometers are the sensor type used the most in fall detection systems before 2014 (Xu *et al.*, 2018). Researchers prefer using this technology owing to its low cost, its portability and because it does not cause privacy concerns. Accelerometers have proven efficient enough to detect falls with high accuracy. However, they have some limitations. This approach requires the subject to wear the sensor all day, which is not only uncomfortable but also affects his or her daily life.

The second category includes WI-FI and radar. This method is based on tracking signal fluctuations to detect fall events. (Tian, Lee, He, Hsu & Katabi, 2018) used Frequency-modulated continuous-wave radio (FMCW) equipped with two antennas to collect radio frequency (RF) reflections from the environment. Two heat-maps were generated from these reflections and used to train a CNN model. This method reached a sensitivity of 94% and a precision of 92%.

RF sensor-based is a non-intrusive method and has achieved a good accuracy level in previous works. However, it is sensitive to interference with other RF-based devices. In our case, it will be challenging to put an antenna in each cell with no risk of interference.

The last category includes RGB cameras and depth cameras. RGB cameras were adopted in previous vision-based methods. In (Chua, Chang & Lim, 2015), authors used RGB images to detect fall events through the use of human shape. After foreground extraction, they represented the human shape by three points (head, body and legs) and calculated the features related to fall. The fall event is detected by comparing these features with a threshold. They achieved an accuracy of 90.5%. Methods based on traditional camera are not robust and their performances are limited due to the difficulty to distinguish the foreground from the background in RGB images. Moreover, they are computationally expensive. The main drawback of these cameras is their sensitivity to illumination variation (Xu *et al.*, 2018).

With depth cameras, some limitations of RGB cameras can be overcome. In addition to an RGB images, these cameras provide a depth image in which each pixel represents the distance of the corresponding object to the camera. Different depth cameras have been released in the past few years, such as Microsoft's Kinect, Intel's Real Sense, and Asus Xtion's Pro Live. Kinect v2 is the successor of Kinect v1 and was released in July 2014. It has become the most popular sensor in fall detection systems (Xu *et al.*, 2018). It is composed of an RGB camera, infrared (IR) camera based on ToF (time-of-flight) technology and a microphone array. Kinect V2 provides different data streams : RGB image, depth map, IR image, skeleton data and audio.

Most vision-based systems use skeleton and joint trajectories of human bodies to detect human daily activities because they are robust to illumination change and scene variation (Yan *et al.*, 2018).

(Gasparrini *et al.*, 2015) opted for a data fusion approach where they used both skeleton data from Kinect v2 and acceleration data from a wearable accelerometer to get higher accuracy. The fall is detected with 99% accuracy by exploiting the variation in the skeleton joint positions, distance of spine base joint to the floor and the acceleration magnitude measured by the waist accelerometer. The authors opted for a threshold-based method. TST v2 dataset is used to evaluate this approach. The results are very good on this dataset but the threshold method is not robust to be generalized since thresholds themselves depend on the subject's characteristics such as sex and age. (Ramachandran & Karuppiah, 2020).

Using both a camera and an accelerometer is more complex then using one sensor since they need to be synchronized. Moreover, sensors be positioned on the body, paying special attention to their orientation, which makes them impractical in our case.

In (Yao *et al.*, 2019), the authors proposed a method that combines a feature-based method and SVM. They used five features calculated from four skeleton joint datasets (head, neck, spine center and spine base). The benefit of this method is that the threshold is learned automatically by the algorithm. They obtained an accuracy of 93.56% on the TST v2 dataset. Despite the use of automatically calculated thresholds, the accuracy remains limited by the choices of features as compared with deep learning methods.

(Tsai & Hsu, 2019) authors decided to extract the important skeleton joints related to falls from the depth image instead of using the joint data provided by Kinect. They then used a 1D convolutional neural network (CNN) to classify the action, achieving an accuracy of 99.2%. This is the first publication that uses the NTU-RGB+D dataset to evaluate a fall detection method. Although this dataset is well-known in the domain of activity recognition, it is not recommended for fall detection. It has some limitations, one of which is that it does not cover all types of falls (Mastorakis, 2018).

In (Le *et al.*, 2014), the skeleton data provided by Kinect were used to calculate the plane corresponding to the floor of the room. Then the authors calculated the distance and velocity of the head and spine relative to the floor. Finally, they used SVM for classification. They got a 0% false positive rate and 3.3% false negative rate which is a very interesting result. The dataset used in this work is not public so we cannot verify the results, and the author mentioned that he removed some fall-like samples (sit and lie down the ground) to obtain better results.

Most of the previous works used a hand-crafted feature, as this does not require a large dataset and allows to explicitly focus on features that largely contribute to the specific task. However, it may be challenging to come up with such reasonable and consistent features on some tasks that require a deep knowledge of the domain. On the other hand, we have the deep learning method, which outperforms the hand-crafted feature extraction method since it can obtain more and more features via the training, but it requires much more data which presents a second limitation.

This paper adopts the skeleton-based model ST-GCN to detect fall events using the skeleton data provided by Kinect v2. We aim to provide a robust fall detection system, and unlike earlier works, we will test our method on different public datasets to demonstrate its generality.

This paper is organized as follows : we describe the proposed system in section 2 and present the results in section 3, followed by a discussion in section 4 and our conclusions in section 5.

2.3 Methods

We propose to work with the Spatial-Temporal Graph Convolutional Network (ST-GCN) introduced in (Yan *et al.*, 2018). The ST-GCN can automatically capture the pattern in the spatial configuration of skeletal joints as well as their temporal dynamics. This algorithm is an extension of the Graphical Convolutional network (GCN), a type of neural network designed to work directly on graphs and exploit their structural information. The main advantage of the ST-GCN is that it can work with data in their native form. We no longer need to reduce the dimensions of graphs to be digestible for machine learning algorithms designed for Euclidean data, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Those machine learning algorithms require a pre-processing step to encode the graph data in vector space. This step can lead to structural information loss from the graph (Dutta, Riba, Lladós & Fornés, 2019). The ST-GCN will help us extract more complex patterns while overcoming the difficulties of encoding temporal information.

In addition to a good model, we need a good dataset, and a good dataset should include a representative and sufficient number of examples to adequately represent the variability of the action, the human subjects and camera views. The existing skeleton-based fall datasets (TST v2 and FallFree) suffer from a lack of large-scale training samples. Although deep learning is one of the most popular machine learning algorithms, to date it has only been adopted in a few fall detection systems (Xu & Zhou, 2018) (Hwang *et al.*, 2017) (Fakhrulddin, Fei & Li, 2017). The main reason is the size limitation of the available fall detection datasets, which limits the performance of deep learning. Transfer Learning can be used to overcome this limitation of data insufficiency.

Transfer Learning is widely used in computer vision where the pre-trained model is used as a starting point for a model on a new task. This technique is inspired by the human ability to reuse their knowledge to learn something new. It is a great tool to handle the problem of insufficient data. In a traditional approach, deep learning algorithms are designed to solve a specific task. These algorithms then need to be retrained from scratch when the task changes. Let us define the task of human action recognition, T1, which has enough data. We train a model until it generalizes well. Now we want to detect fall events (task T2) with significantly fewer data. We can apply the model already trained on T1 or we can retrain it from scratch on T2. In the first case, we observe a performance degradation due to the unbalanced dataset used in T1. In the second case, we get a poor performance, since the dataset for fall detection is not large enough for generalization. Transfer learning enables us to use the knowledge of T1 (features and weights) for T2. The idea is to freeze the first layers of the network, so the weights remain unaltered during the training on T2. Freezing the first layers allows them to then learn the features shared across the two tasks. The benefit of using transfer learning is to learn the general features of human activity and then use the fall class dataset to learn the specific features of our task.

Our approach to develop an efficient fall detection technique utilizes the ST-GCN to extract complex patterns combined with the transfer learning technique.

2.3.1 Architecture of the network

The ST-GCN shown in figure 2.3 is composed of 10 layers of spatial-temporal convolution (ST-GCN units). The first four layers have 64 output channels, the next three layers each have 128 output channels and the last three layers have 256 output channels. The convolution kernel size is set to 9. To avoid overfitting, two techniques have been used : the residual network mechanism and a drop-out layer with a 0.5 drop rate. Finally, we feed the resulting feature vector to a Softmax classifier. The model is learned using a stochastic gradient descent (SGD) optimizer and a learning rate of 0.01.

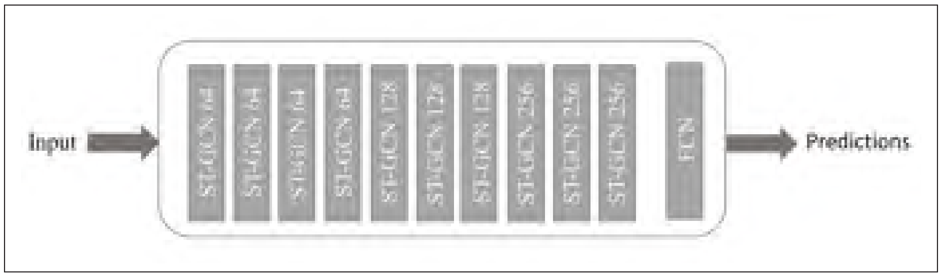


Figure 2.3 Spatial temporal graph convolution network architecture

The input data is presented as a graph which is composed of nodes and joints as illustrated in figure 2.4. Each joint represents a node of the graph. We have 2 types of edges : inter-body edges and intra-body edges. The first type represents the anatomical connections in the human body and the second connects each node to itself in two consecutive frames.

2.3.2 Dataset

We selected 3 datasets for this work : the NTU-RGB+D (Shahroudy *et al.*, 2016), the TST v2 (Enea *et al.*, 2016) and Fallfree (Alzahrani *et al.*, 2017). Table 2.1 summarizes the features and specifications of each dataset including the number of subjects, actions, fall samples, ADL samples and the year of creation.

Table 2.1 RGB-D fall datasets

Dataset	**Subjects**	**Actions**	**Fall samples**	**ADL samples**	**Year**
NTU-RGB+D	40	60	276	55724	2016
TST v2	11	5	132	132	2015
Fallfree	2	10	208	183	2017

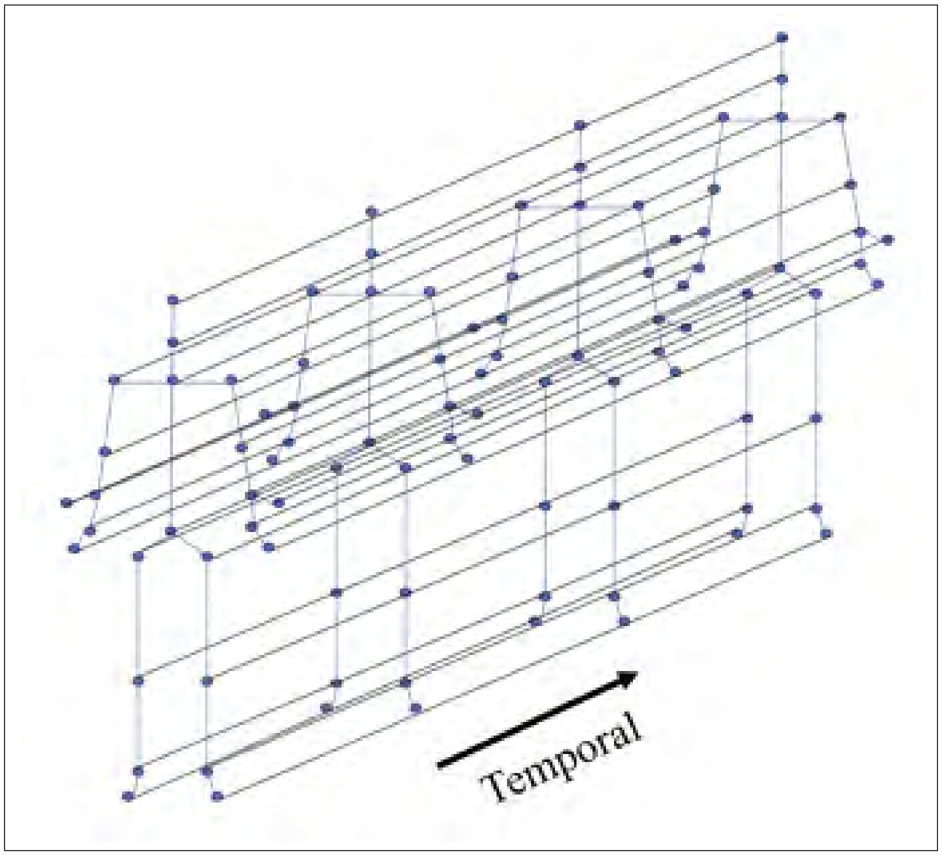


Figure 2.4 The spatial-temporal graph of skeleton sequence ; inter-body edges (blue) and intra-body edges (black)

The NTU-RGB+D dataset was collected using a Kinect v2 camera and published in 2016. It contains 56,000 action samples divided into 60 classes. Four data modalities are provided for each sample : RGB frames, depth maps, skeleton data and IR sequences. The actions depicted in the samples were performed by 40 subjects who are between 10 and 35 years old. Three cameras were used at the same time to capture three different views for the same action. Each subject repeats the action twice.

The TST v2 dataset was published in 2015. It was collected using a Kinect v2 camera and two accelerometers placed on each actor's waist and wrist. This dataset is composed of 264 samples divided into two main groups : ADL and Fall, each containing 132 samples. The ADL group contains information on the following actions : sit, grasp, walk and lay, while the fall group contains information on the following actions : front fall, backward fall, lateral fall and ends up sitting fall. The data provided in this dataset are the depth frame, the skeleton joints and accelerometer data. The actions were performed by 11 subjects between 22 and 35 years old with different heights (1.62-1.97 metre). Each subject repeats each action three times.

The FallFree dataset was collected using a Kinect v2 camera facing the subject, and was published in 2017. It contains 391 samples divided into three main categories : true fall with 208 samples, pseudo-fall with 115 samples, and ADL with 68 samples. The true fall category covers the following actions : front fall, backward fall and lateral fall. The pseudo-fall category contains falls with recovery. A fall with recovery occurs when the person loses their balance but then recovers, regaining their balance. The ADL category contains neutral actions such as : sit, stand up, lie, etc. The data are provided in the form of several video clips (eXtended Event File - XEF). Each XEF contains five data streams : RGB frames, depth, skeleton joints, IR sequences and body index. These actions were performed by two subjects in three different rooms and three different lighting conditions. The first subject, who was 30 years old and 1.50 metres tall, repeated each action 4 to 5 times, and the second subject, 35 years old and 1.68 metres tall, performed each action one time. Both subjects used a cane in 75% of the samples.

The NTU-RGB+D dataset appears to be the benchmark source in the action recognition domain. This dataset is characterized by its variability of action, number of human subjects, camera views and environmental conditions. Despite its characteristics, this dataset was not oriented towards fall detection. Subjects all perform the same type of fall, which can overfit our model. Moreover, the fall did not always conclude with lying on the floor (Mastorakis, 2018). We cannot expect to have a generalizable method when this dataset is used for training. However, it represents the best option for testing the generality of our method and how our model behaves on a new dataset.

Unlike the first dataset, TST v2 and Fallfree were prepared for fall detection studies. They contain different types of falls and ADL actions. In addition, they have been used to evaluate fall detection algorithms (Gasparrini *et al.*, 2015) (Fakhrulddin *et al.*, 2017) (Hwang *et al.*, 2017) (Maldonado-Mendez, Solis, Rios-Figueroa & Marin-Hernandez, 2017) (Min, Yao, Lin & Liu,

2018) (Seredin, Kopylov, Huang & Rodionov, 2019) (Maldonado-Mendez & Hernandez-Mendez, 2019) (Yao *et al.*, 2019) (Alzahrani, Jarraya, Ben-Abdallah & Ali, 2019). Both of these aspects make them a good option with which to evaluate our method.

The TST v2 dataset involves more subjects (11) than Fallfree (2), which helps generalize our model. Unlike the TSTv2 dataset, the FallFree dataset includes pseudo-fall actions such as falls with recovery, which is more challenging for fall detection systems. FallFree is in the form of XEF files and contains all the data streams provided by a Kinect v2 camera. Therefore, we need an extra tool to extract the skeleton joints. The TST v2 dataset offers depth and skeleton data as well as accelerometer data. Unlike the TST v2 dataset, Fallfree was recorded in three different rooms with three different lighting conditions.

Given that each of these datasets has their advantages and limitations, we decided to use all of them to train our model. Our goal is to build a generalized fall detection system and to improve its ability to classify new data. We use the transfer learning technique to take advantage of the size and variability of the NTU-RGB+D dataset, using it to train our model from scratch. Next, we use TST v2 and Fallfree to perform a fine-tuning and obtain our final model. This process will increase the performance of our method.

2.3.3 Training

Our method relies on two-step training processes to train our model for fall detection as shown in figure 2.5 :

- we train the ST-GCN network from scratch using the NTU-RGB+D dataset for 100 epochs with the same configurations and parameters proposed by (Yan *et al.*, 2018). This dataset contains 56000 video samples and 60 action classes. Every video has at most two subjects in the scene and it is represented as a tensor of (3,T,25,2) dimension ; with :
  - 3 is the 3D joint coordinates (X,Y,Z) ;
  - T is the number of frames in one video clip ;
  - 25 is the number of skeleton joints for each subject ;

- 2 represents the number of subjects in the same clip.

The output layer of the model is a vector of 60 elements (the number of classes). This vector represents the probability that a given input belongs to a class (Di Giamberardino, Iacoviello, Tavares & Jorge, 2012).

During this step, the model learns the characteristics that represent general human movement.

- based on the pre-trained model, we froze the first nine convolution layers, so their weights remain unchanged during back-propagation. The input of the model remains the same. The output layer dimension is changed to the new number of classes, in our case 2. We then perform a new training, based on the TST v2 and Fallfree datasets. We split the data according to the cross-subject evaluation method. This evaluation method is dicussed in detail in the next section (3.A).

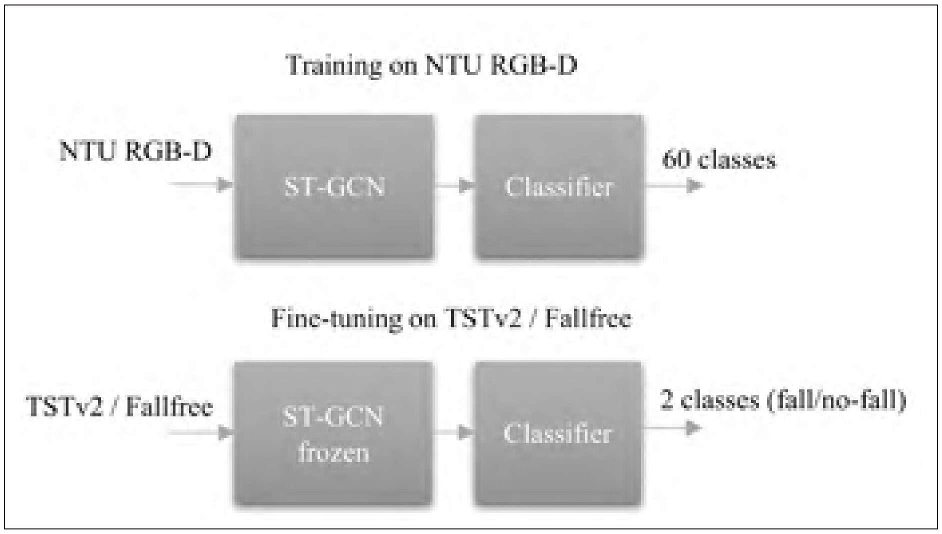


Figure 2.5 Transfer learning process

2.4 Results

To evaluate our method, we conducted many experiments using three public datasets : NTU RGB-D, TST v2 and FallFree. It is not straightforward to make an objective comparison between the different methods proposed in the literature because they are generally evaluated on different datasets. These datasets are also acquired with different configurations and they involve a different number of participants and actions. Therefore, we decided to compare our results with works that have used the same datasets.

2.4.1 Training and Evaluation Protocol

Unlike TST v2, Fallfree proposes an evaluation protocol that leads to an unfair comparison in our opinion. It uses only samples from the fall classes which leads to ignoring the false-positive cases. To avoid this biased evaluation, we use the cross-subject protocol, the most commonly-used protocol in this field (Wang *et al.*, 2018).

The TST v2 datastet comprises videos performed by eleven subjects. We split the data among a training group comprised of data performed by seven subjects and a testing group comprised of data performed by the remaining four subjects. The training subjects are randomly selected as follows : subjects 1, 3, 5, 7, 9, 10 and 11, and the remaining subjects were used for testing. The Fallfree dataset comprises data performed by only two subjects : the first subject performs each action six times, and the second subject performs the actions only once. Therefore, we use the first subject to train our model and the second subject to test it.

We report here on six experiments in order to compare our methods within different situations and to evaluate its robustness.

- Experiment 1 : We train our model utilizing the NTU-RGB+D dataset, and then apply transfer learning on the TST v2 dataset using samples recorded by seven subjects (subject's IDs : 1, 3, 5, 7, 9, 10, 11). Finally, we test the model based on the TST v2 dataset using samples recorded by the four subjects that were not used for training (subjects' IDs : 2, 4, 6, and 8).
- Experiment 2 : We train our model utilizing the NTU-RGB+D dataset, and then apply transfer learning based on the TST v2 dataset using samples recorded by seven subjects (subject's IDs : 1, 3, 5, 7, 9, 10, 11). Finally, we test the model on the entire FallFree dataset.
- Experiment 3 : We train our model utilizing the NTU-RGB+D dataset, and then apply transfer learning based on the FallFree dataset using samples recorded by the first subject. Finally, we test the model on the Fallfree dataset using samples recorded by the second subject.

- Experiment 4 : We train our model utilizing the NTU-RGB+D dataset, and then apply transfer learning based on the FallFree dataset using samples recorded by the first subject. Finally, we test the model on the entire TST v2 dataset.
- Experiment 5 : We train our model utilizing the NTU-RGB+D dataset, and then apply transfer learning based on a new dataset (Fall-train). This dataset was created using :
  - samples recorded by seven subjects from the TST v2 dataset (subject's ID : 1, 3,5, 7, 9, 10, 11) ;
  - samples recorded by the first subject from the FallFree dataset.

  Finally, we test the model on a new dataset (Fall-test), created using :
  - samples recorded by four subjects from the TST v2 dataset (subject's IDs : 2, 4, 6 and 8) ; and
  - samples recorded by the second subject from the FallFree dataset.
- Experiment 6 : We train our model from scratch based on the Fall-train dataset. Then, we test the model on the Fall-test dataset.

Table 2.2 summarizes the proposed evaluation method for all six experiments.

2.4.2 Evaluation metrics

The problem of fall detection can be seen as a binary classification problem, and the most common metrics used to evaluate such a classifier are :

- Sensitivity/Recall : represents the True Positive Rate =TP/(TP + FN) ;
- Specificity : represents the True Negative Rate =TN/(TN + FP) ;
- Accuracy : represents the Percentage of total items classified correctly =(TP + TN)/(TP+TN+FP+FN),

where :

- TP : represents the True Positives which are the actions labelled as 'fall' and predicted as 'fall' ;
- FP : represents the False Positives which are the actions labelled as 'no fall' and predicted as 'fall' ;

Table 2.2 Training and evaluation protocol

Experiment	**Training data**	**Transfer learning training data**	**Testing data**
1	NTU-RGB+D	7 subjects from TST v2 (subject's ID : 1, 3, 5, 7, 9, 10, 11)	4 subjects from TST v2 that are not used for the training (subject's ID : 2, 4, 6, 8)
2	NTU-RGB+D	7 subjects from TST v2 (subject's ID : 1, 3, 5, 7, 9, 10, 11)	entire FallFree
3	NTU-RGB+D	Subject 1 from Fallfree	Subject 2 from Fallfree
4	NTU-RGB+D	Subject 1 from Fallfree	entire TST v2
5	NTU-RGB+D	7 subjects from TST v2 (subject's ID : 1, 3, 5, 7, 9, 10, 11) and Subject 1 from Fallfree	4 subjects from TST v2 that are not used for the training (subject's ID : 2, 4, 6, 8) and Subject 2 from Fallfree
6	7 subjects from TST v2 (subject's ID : 1, 3, 5, 7, 9, 10, 11) and Subject 1 from Fallfree	not applied	4 subjects from TST v2 that are not used for the training (subject's ID : 2, 4, 6, 8) and Subject 2 from Fallfree

- TN : represents the True Negatives which are the actions labelled as 'no fall' and predicted as 'no fall' ;
- FN : represents the False Negatives which are the actions labelled as 'fall' and predicted as 'no fall'.

2.4.3 Results

The results of our experiments are summarised in table 2.3.

The first experiment gives an excellent result when training and testing on the TST v2 dataset. We reach an accuracy of 100%, a recall of 100%, a specificity of 100% and a 0% FP rate.

The second experiment gives an acceptable result when tested on Fallfree. We obtain a 78.26% accuracy with a balanced rate of sensitivity (71.63%) and specificity (85.8%), and a low FP rate of 14.2%. These results are good, especially since no data from the Fallfree dataset was used in the training and thus the complete Fallfree dataset is unseen by our model.

The third experiment produces a very good result with training and testing on a Fallfree dataset. We obtain a very good accuracy of 97.33%, a recall of 97.5%, a specificity of 97.14% and a very low FP rate (2.86%).

The fourth experiment shows an acceptable result with testing on the TST v2 dataset. We get an accuracy of 75.38%. We note that this setup results in 100% rates of sensitivity which means that our method has detected all the fall events, but with a high false positive rate of 49.24%. So, we can conclude that our method learns better from the TST v2 dataset than from the Fallfree dataset.

In the fifth experiment, combining the two datasets, we get a 100% accuracy and all events are classified correctly. This proves that our method could be generalized quite easily.

In the sixth experiment, we train our model from scratch on both datasets. We get a poor result (64.58% accuracy) compared to the fifth experiment that combined both datasets. By comparing the results of these two experiments, we can conclude that transfer learning improves the performance of the model. Using transfer learning with the same configuration and data split, we could increase the accuracy by 35.42% and decrease the FP rate by 70.83% when comparing the results between experiment five and experiment six.

Table 2.3 Results of our fall detection method

Experiment	Accuracy	Recall/sensitivity	Specificity	FP rate
1	100%	100%	100%	0%
2	78.26%	71.63%	85.8%	14.2%
3	97.33%	97.5%	97.14%	2.86%
4	75.38%	100%	50.76%	49.24%
5	100%	100%	100%	0%
6	64.58%	100%	29.17%	70.83%

2.4.4 Comparison with state-of-the-art

To our knowledge, we are the first to combine TST v2 and Fallfree datsets for training and testing a fall detection system. Therefore, we will compare our approach with the state-of-the-art on each dataset separately. We select the methods that used the TST v2 dataset and the Fallfree dataset to compare with our proposed system. We compare our approach to that of others with respect to accuracy, because not all the methods described in the literature use the other metrics. Unfortunately, the comparison depends upon the size of the training and testing datasets and especially on the choice of the samples in the testing phase ; in other words, the evaluation method. A detailed comparison with nine different methods is presented in table 2.4 and table 2.5 ; where each row summarizes the comparison results with a specific referenced work. For each comparison, we describe the dataset used for training, the algorithm used for training, the validation method and the classification result in terms of accuracy.

The best accuracy achieved in previous works is 99% in (Enea *et al.*, 2016). It is explained by the usage of both skeleton and accelerometer data. The next-highest accuracy, 94.2% reached by another study that used only one type of data in (Kouyoumdjian *et al.*, 2016).

2.4.4.1 Comparison with results using the TST v2 dataset

The TST v2 dataset is used more than FallFree and the NTU-RGB+D dataset in the area of fall detection. Therefore, we accorded more importance to it and so aimed to include all the fall detection works that use the TST v2 dataset in table 2.4.

Among these studies, (Gasparrini *et al.*, 2015) reached the highest accuracy, 99%, as it used both accelerometer and skeleton data to calculate the variation in the skeleton joint position, the distance of the spine base joint from the floor and the acceleration magnitude measured by the waist accelerometer. They exploited these parameters to detect fall events using a threshold-based method.

In (Fakhrulddin *et al.*, 2017), the authors applied a CNN to classify fall and ADL actions. They converted data collected from 2 accelerometers into two images (one image for each accelerometer's data) to be digestible by the CNN. They split the dataset randomly into a training set (90%) and a test set (10%) and then they fed it to the CNN. They used ten-fold cross validation and reached an accuracy of 92.3%.

Hwang et al. (Hwang *et al.*, 2017) applied a data augmentation technique on the depth map data to prevent overfitting. Next, they fed the data to a 3D Convolutional Neural Network (3D-CNN). They used five-fold cross-validation. This work obtained a good accuracy of 92.4%~96.9%, with a mean of 94.2%.

(Maldonado-Mendez *et al.*, 2017) combined a depth map and skeleton joints' positions. They decided to exclude the grasp action from the dataset and thus used only seven actions. In order to select the most relevant features, a Genetic Algorithm (GA) was used to select 37 features. A GA is a search heuristic method that simulates the processes of natural selection. This technique is especially useful when the search space is large. Next, they split the data into two sets. The first set is composed of 14 samples (2 sequences of each action) for feature selection and to fit the parameters of the generative model. The second set is composed of 231 samples, utilised to evaluate the model. The goal of the generative model is to build a probability model of the training data and to generate new samples from the same distribution. This work achieved the lowest accuracy, 81.81%.

(Min *et al.*, 2018) selected 16 skeleton joints from 25 joints provided by Kinect v2. For each joint, they calculated the "minimum height of the given joint" and the "max joint vertical velocity".

Then they trained an SVM model using two-thirds of the data. The other one-third was used to evaluate the method, realising an accuracy of 92.05%.

In (Seredin *et al.*, 2019), the authors decided to use only 17 skeleton joints. First, they calculated the Euclidean Distance Matrix (EDM) and inter-frame speed for each of the 17 joints and obtained 459 features. Next, they used the SVM one-class classifier to detect and eliminate the skeleton joint errors that were not properly estimated. They then trained a two-class SVM and calculated the distance from the hyperplane. Finally, they used the Cumulative Sum (CUSUM) method to detect fall events. This method is based on the calculation of the cumulative sum of a statistical characteristic. It is used to monitor changes in the input data. In their work, the characteristic is the distance from the hyperplane. They used leave-one-person-out cross-validation as an evaluation method and reached an accuracy of 91.7%.

In (Maldonado-Mendez & Hernandez-Mendez, 2019), the authors decided to use only seven actions and to exclude the grasp action. They calculated the speed and acceleration in 20-frame windows, and then selected eight features using a GA. Fourteen samples (two sequences of each action) were used for feature selection, and the remaining 217 (31 sequences of each action) samples were used to train and test the model. They used the SVM model to detect fall posture. If a fall posture was detected, they calculated the velocity and acceleration of the right wrist, the left wrist and the centroid (center of the point cloud). They then compared these features with a threshold value. If the velocity and acceleration of two of these elements were over 1000, a fall was detected. They reported an accuracy of 90%.

In (Yao *et al.*, 2019), the authors used four skeleton joints : head, neck, spine center and spine base to calculate five features : Minimum Head Height, Minimum Centroid Height, Maximum Head Vertical Velocity, Maximum Centroid Vertical Velocity, and Maximum Torso Angle Increasing Velocity. Then they used the SVM algorithm to classify the action as fall or not fall. The dataset was divided into a training set (70% of the data) and a test set (30% of the data). They reported an accuracy of 93.56%.

Table 2.4 summarizes the comparisons between previous works and our own. Our proposed method outperforms all of the methods that used the TST v2 dataset. We achieve an accuracy of 100% even though we use a challenging evaluation method that trains the model on some subjects and tests it on the rest of the subjects.

2.4.4.2 Comparison with results using the FallFree dataset

Fallfree has not been used in many studies to validate methods as it is a very recent dataset (2017). (Alzahrani *et al.*, 2019) examined 11 sets of features used in previous works (Kawatsu, Li & Chung, 2013) (Lee & Lee, 2013) (Le *et al.*, 2014) (Kwolek & Kepski, 2014) (Alzahrani, Jarraya, Ali & Ben-Abdallah, 2017), aiming to determine which subsets of features are the most relevant to fall detection systems. These features were calculated from the skeleton joints provided by Kinect v2 cameras. The researchers decided to use a non-parametric method that uses neighborhood component analysis (NCA) to select featuress. They split the dataset into two subsets : a training set (70%) and a test set (30)%, and then evaluated these sets with various classifiers to determine the most appropriate one. The Random forest model (RF) gave the best performance and achieved an accuracy of 99.95%. We compare their results with ours in table 2.5. We did not use the same validation protocol. For example, in (Alzahrani *et al.*, 2019) the authors used the same subjects for training and testing. In our case, we use one subject for training and a different one for testing, which is more challenging. This difference can explain why we report a slightly inferior accuracy.

2.4.4.3 Comparison with results using the NTU-RGB+D dataset

NTU-RGB+D is an action recognition oriented dataset. It contains 60 different actions with only one fall action, which makes it a realistic dataset that can be used to simulate real-life scenarios.

In (Tsai & Hsu, 2019), the authors used the NTU-RGB+D dataset to train and evaluate their method. They extracted skeleton data from depth data instead of using skeleton joints provided by Kinect v2, selecting seven joint points which they found relevant to fall detection. The

Table 2.4 Comparison between our method and other state-of-the-art methods tested on the TST v2 dataset

Method	**data**	**algorithms**	**Evaluation method**	**Accuracy**
(Gasparrini *et al.*, 2015)	Skeleton joint and Accelerometer	threshold-based algorithm	Not mentioned	99%
(Fakhrulddin *et al.*, 2017)	accelerometer	CNN	90% for training and 10% for testing. Repeat splitting and training. Then averaging	92.3%
(Hwang *et al.*, 2017)	Depth map	3D-CNN and data augmentation	5 random trials of : 240 (for training) and 24 (for testing) videos. Then averaging	94.2%
(Maldonado-Mendez *et al.*, 2017)	Depth map and Skeleton joints	Feature selection using a GA and a generative model	2 sequences of each action were used for Feature selection and training. 231 sequences for evaluation	81.81%
(Min *et al.*, 2018)	Skeleton joints	SVM	2/3 of data for training and 1/3 for testing	92.05%
(Seredin *et al.*, 2019)	Skeleton joints	SVM and CUSUM	Leave-one-person-out	91.7%
(Maldonado-Mendez & Hernandez-Mendez, 2019)	Skeleton joints	Feature selection using a GA, SVM and a threshold model	2 sequences of each action were used for Feature selection and 217 sequences for classification	90%
(Yao *et al.*, 2019)	Skeleton joints	SVM	70% for training and 30% for testing Accuracy calculated on the whole dataset	93.56%
Our method	Skeleton joints	ST-GCN	Cross subject	100%

Table 2.5 Comparison between our method and other state-of-the-art methods tested on the Fallfree dataset

Method	**data**	**algorithms**	**Evaluation method**	**Accuracy**
(Alzahrani *et al.*, 2019)	Skeleton joint	Feature selection using NCA and Random forest	70% for training and 30% for testing	99.5%
Our method	Skeleton joints	ST-GCN	Cross subject	97.33%

NTU-RGB+D dataset contains 60 classes, therefore they considered all falling samples as the positive class and then randomly selected samples from other classes to build the negative class. Next, they split their data into a training set (70%) and a testing set (30%) and fed them to a 1D-CNN.

This is the only study that used NTU-TGB+D to study fall detection, and so we consider this work with which to compare our results. We decompose the NTU-RGB+D dataset into fall and no-fall classes, as mentioned in (Tsai & Hsu, 2019) and we reduce the number of no-fall samples to obtain a balanced dataset. We do not try to retrain or fine-tune our model on this dataset ; we will only test the unseen data to determine how our method reacts to the problem of a cross-subject dataset. The authors reached an accuracy of 99.2% with a model already trained a different part of on the same dataset, NTU-RGB+D. In our case we achieved an accuracy of 92.91 % with unseen data as shown in Table 2.6. Given that our results are obtained with entirely unseen data, we consider that we have achieved very good results.

2.5 Discussion

Results obtained with the TST v2 dataset are better than those obtained with the Fallfree dataset. This is observed by comparing experiments 1, 2 , 3 and 4. The models trained on TST v2 gives better and more balanced TP, TN and FP rates on unseen data than the one trained on Fallfree. The model trained on the Fallfree dataset and described in experiment 4 suffers from overfitting,

Table 2.6 Comparison between our method and other state-of-the-art methods tested on the NTU RGB-D dataset

Method	**data**	**algorithms**	**Evaluation method**	**Accuracy**
(Tsai & Hsu, 2019)	depth data	CNN	70% for training and 30% for testing	99.2%
Our method	Skeleton joints	ST-GCN	Cross subject	92.91%

which is why it has a high FP rate. This model gives more importance to the "fall" class. These results can be explained by the limited variability of subjects in the FallFree dataset.

In the fourth experiment, we get a poor accuracy with a very high FP rate. This is due to insufficient data.

From our experiments we can conclude that using transfer learning with a combined dataset for training provides significantly better results. This helps to generalize the model and prevents the overfitting problem. On the other hand, the combination of the TST v2 and the Fallfree datasets contributes to overcoming their specific limitations and to maximizing the results. These two datasets can be complementary ; the TST v2 dataset has more subjects than Fallfree, but fewer actions. Fallfree is more challenging because it contains pseudo-fall events like syncope and fall with recovery. These two datasets suit our case study since the environments where the actions were recorded are similar to prison cells.

We realized an additional experiment on the NTU RGB-D dataset to test the generality of our method : This experiment presents two major challenges.

- first, as the dataset is totally unseen by our model, the difference of actors and view angles may cause distribution divergence, which could affect the accuracy.
- second, our model is trained on two datasets that contain one fall action and at most ten different ADL actions. On the other hand, NTU RGB+D contains one fall action and 59 ADL actions. This means we have 49 new actions that our model must differentiate from falling.

Despite these constraints, the results were very good.

Unfortunately, neither of these fall datasets contain occlusion scenarios, which limits the generability of our model. Occlusion is a well-known problem in human action recognition and it is inevitable in real life. To work around this problem, we can use two synchronized Kinect cameras in the same cell.

2.6 Conclusion and future works

In this paper we propose a general fall detection system which could be applied directly on different datasets. Our method is based on transfer learning from action recognition, applied to fall detection. Our model exploits skeleton data provided by the Kinect v2 camera and can detect up to two persons in the same scene. We use three public datasets : NTU-RGB+D, TST Fall detection v2, and Fallfree. Our method outperforms the previous state-of-the-art methods on different benchmark datasets and proves its robustness and generality.

We achieve such good results in detecting fall events mainly because :

- we used RGB-D camera and especially skeleton data, which are environment- and lighting conditions-independent ;
- we used ST-GCN, which can detect the spatial-temporal correlation of graphic data, in our case the skeleton data ;
- we used transfer learning to overcome the problem of a lack of data ; and
- we used two different fall datasets to generalize our model.

Our proposed method provides a non-intrusive system with high precision that is robust to environmental changes due to the use of skeleton data.

However we have used simulation data, which has its own limitations. Hesitation is considered to be an important element related to unrealistic falls (Mastorakis, 2018). Falling is a dangerous action, and hesitation occurs due to the fear of an injury. Involuntarily, the subject tries to minimise the impact of a fall or to avoid an injury.

We could not test how our approach reacts to occlusion scenarios with the presence of objects that obstruct skeleton data, such as furniture. In an indoor scene, especially in a home, people generally move around furniture and so partial or total occlusion scenarios will occur. This makes the available fall dataset unrepresentative of real life.

In our future works, we will focus on detecting other symptoms of overdose and apply our method on real-time detection.

Acknowledgment

This paper and the research behind it would not have been possible without the support of Natural Sciences and Engineering Research Council, Aerosystems International Inc., and Prompt Quebec and Mitacs.

CHAPITRE 3

REAL-TIME SKELETON BASED FALL DETECTION SYSTEM

Oussema Keskes[1] , Rita Noumeir[2]

[1] Département de génie logiciel et des technologies de l'information, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

[2] Département de Génie Électrique, École de Technologie Supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article soumis à la conférence « Conference on Computer Vision and Pattern Recognition » en octobre 2020.

3.1 Abstract

During the last decade, real-time surveillance has become a hot topic especially with the appearance of the RGB+D cameras. In this paper we focus on detecting falls. Fall can be caused by overdose or unconsciousness due to a health problem. In both cases, an immediate intervention is needed. Most previous works proposed an offline model which requires additional processing to be able to detect fall in real-time. In this work we propose a real-time fall detection system based on ST-GCN and 1D-CNN models using skeleton data. Our method proves its robustness and generality by using different datasets for the training and for testing phase. We use PKU-MMD dataset to evaluate our method. We obtain an accuracy of 90% in real-time and we succeed to detect all fall events.

keywords - Suicide preventing, Unconsciousness, Fall detection, Deep learning, ST-GCN, RGB-D data, skeleton, TST Fall detection dataset v2, Fallfree, NTU RGB-D

3.2 Introduction

Overdose are an increasing problem in the Canadian correctional services (Kouyoumdjian *et al.*, 2016). A fall usually occurs when a person loses his/her consciousness due to an overdose. A real-time fall detection system is needed to provide an immediate intervention.

Previous fall detection systems can mainly be divided into two categories :

- wearable sensor based ;
- environmental sensor based.

The first category includes accelerometers, gyroscopes, and other wearable sensors. They calculate the variation of human posture like the velocity of the body and physical state such as the heart rate. For example, in (Wang *et al.*, 2014), using a 3D-accelerometer, the authors calculated the Sum Vector Magnitude of Acceleration and the trunk angle. An additional pulse pressure sensor was used to calculate the heart rate. If these features exceed a threshold, an alarm is triggered. This approach has the advantage to be low cost and non-intrusive. On the other hand, it can be uncomfortable to wear a device all day and an incorrect positioning of the sensor can significantly affect the accuracy of the system.

The second category includes sensors that monitor changes in a specific space such as RGB cameras, RGB-D cameras and Radio frequency (RF) sensors. Before the appearance of the RGB-D camera, accelerometers were the most used sensors for fall detection (Xu *et al.*, 2018).

Traditional environmental sensor-based methods mainly rely on the RGB camera. This sensor has the advantage of being low cost and easy to install. For example, (Chua *et al.*, 2015) extracted the human body shape from RGB images and calculated features related to fall such as orientation and heights. A threshold-based method was used to detect a fall by comparing these features with a predetermined value. This approach is computationally more costly than wearable sensor-based methods and sensitive to lightning changes.

Radio frequency (RF) sensors include WI-FI and radar. It is based on signal processing to detect the fall. For example, (Su, Ho, Rantz & Skubic, 2014) used Wavelet transform (WT) methods to detect falls. They calculated the discrete Stationary Wavelet Transform (SWT) coefficients using the gated sinusoids provided by a Doppler radar. The nearest neighbour (NN) algorithm was used to distinguish fall from activity of daily living (ADL) actions. The main drawback of these sensors are their limited coverage (Ren & Peng, 2019) and their sensitivity to interference.

RGB-D cameras attracted many researchers and became the mainstream in fall detection systems. (Min *et al.*, 2018) used the skeleton data provided by the Kinect v2 camera to calculate features related to fall such as velocity. The fall were detected using a Support Vector Machine (SVM) classifier.

Most previous studies addressed fall detection as an offline problem where the input data are a well-segmented clip. These methods are trained and tested using segmented sequences, where each sequence contains only one action. The model provides the classification result after analyzing the whole sequence. In real life, the model will receive a continuous stream of data containing different actions with unknown start and end times. first, the model has to detect the start and the end of the action, then it can classify it. The model should perform these two tasks in real-time.

Real-time fall detection is a new challenge. It goes one step further then a traditional fall detection system which detects a fall from a segmented clip. Unlike conventional fall detection systems which observe the entire clip before providing a classification of the action, an real-time fall detector should detect the fall on the fly from a continuous data stream. The fall detection should be performed in an online setting and at real-time speed.

In the fall detection domain, most previous works are threshold-based methods which can be easily applied in real time. On the other hand, applying methods based on machine learning is not straightforward and requires additional work. First, we should verify that the methods are fast enough, then they can be adapted to the real time problem.

Threshold methods are based on comparing features extracted from the collected data with a reference value called a threshold. This value is set after many experiments. Moreover, the main drawback is the choice of the threshold value and the need for hand-crafted features. A high value will miss some fall events, while a low value will cause many false alarms. The success of this approach comes from its low complexity. This approach is mainly used with accelerometers sensors.

For example, (Pham, Dao, Phung, Van Ta, Nguyen & Hoang, 2018) collected data from an accelerometer. Then, they calculated the acceleration changes and the angle of the module compared to the vertical axis. Fall event were detected using a predetermined multi-threshold approach. They reported an accuracy of 92%. The dataset used in this work contains a fall action and six different ADL (activity of daily living) actions.

The threshold-based method has the advantage of being light-weight and can be used in real-time with few modifications. Otherwise, it is not robust, and the choice of the thresholds depends on the subjects used in the experiments. The selected threshold values can work well on some subjects while it can give a low accuracy on others.

With the appearance of new advanced sensors such as RGB-D cameras, machine learning algorithms have become the new mainstream approach to fall detection (Xu *et al.*, 2018). Machine learning based methods prove their robustness and generality compared to threshold based methods, but they require more calculation and some methods cannot be directly used in real-time. While some researchers state that their models are fast enough to work in real-time (Tsai & Hsu, 2019) but few of them are actually implemented in real-time.

For example, (Tsai & Hsu, 2019) proposed a vision-based fall detection method. They extracted skeleton coordinates from the depth image. A CNN algorithm was used to classify the input action and reached an accuracy of 99.2%. Authors mentioned that their method is fast enough and has been implemented in real-time but they did not report the procedure of the implementation or any new results.

(Chenguang *et al.*, 2020) proposed a real-time fall detection method using a monocular camera. They used a Gaussian mixture model (GMM) to detect the foreground. After that, two ellipses were used to fit the head and the torso and they calculated three features for each one : the inclination angle of the ellipse, the velocity and the ration of short and long axis of the eclipse. Finally, they used a Convolutional Neural Network (CNN) to model the correlation between the two ellipse. This work was trained and tested on a dataset collected by the authors and reaches an accuracy of 90.5%. We assume that they used segmented videos since they did not provide a description of the dataset.

(Wu *et al.*, 2019) encoded the skeleton joints into a RGB image. Then, they fed them to a lightweight CNN algorithm. They used a combination of their collected dataset using a Kinect v2 camera and MSRDaily Activity3D (Li *et al.*, 2010) dataset to evaluate their method. They achieved an accuracy of 93.75% in offline mode. A voting method was used to detect falls in real-time. Every set of 15 frames was classified as a fall or non-fall. If three of the last five classifications were classified as a fall, a fall event was detected. They tested their approach using 20 sequences and they reported an accuracy of 95%. The main drawback of this work is that the dataset used contains only segmented videos, in other words each video contains only one action.

(Triantafyllou *et al.*, 2016) proposed a real-time fall detector in indoor work environments. They extracted three features from depth frames : vertical velocity, area variance and the height of the person. The fall is detected using a three state Markov model. The dataset was collected using two Kinect v2 cameras in two different working environments. The fall is detected with 97.14% accuracy. This method relies on hand-crafted features that limit the robustness and the generality of the model.

(Musci *et al.*, 2018) opted for an accelerometer-based method to detect falls. They used the SisFall (Sucerquia *et al.*, 2017) dataset to train a Long Short-Term Memory network (LSTM). They divided all the SisFall sequences into windows of 1.28 s with a partial overlap of 50% and they manually labelled the dataset to fit their model. Their method reached an accuracy of 97.16%. Despite the good results, this work relies on 3D-accelerometer sensors which need to be worn all day. This is uncomfortable and can be altered by the prisoner.

We extend the method proposed in our previous work (Oussema & Rita, 2020) by adding an action detection module. This allows our method to work in real-time. We propose an action detection module based on skeleton data provided by the Kinect v2 camera. Firstly, we train a 1D-CNN algorithm on PKU-MMD (Liu, Hu, Li, Song & Liu, 2017) dataset. Then we test the whole fall detection system which contains an action detection module and a fall detection module, using the same dataset.

The choice of using deep learning, specifically 1D-CNN, is based mainly on two points :

- the need of a robust model that can automatically extract the relevant features ;
- as we propose a real-time fall detection, the model should be light and fast enough to be implemented.

We offer a robust and reliable real-time fall detection system, that is independent from the subject, the camera view and the environment setting.

Our main contributions are :

- implementing a deep learning algorithm to detect the start and the end of the action ;
- applying the ST-GCN algorithm to detect falls as they occur in real-time.
- to our knowledge we are the first to propose a cross-dataset fall detection model. Although we used different datasets for training and for testing, we reach excellent results. The advantage of our method, is that we don't need to retrain the model when we change the dataset.

This paper is organized as fellow : we present the proposed method in section 2, we describe the dataset used in this work in section 3, we describe experiments in section 4, we present results in section 5, discussion in section 6 and conclusion in section 7.

3.3 Methods

We propose a novel online fall detection system using two modules : an action detection module based on a 1D-CNN and a fall detection module based on a ST-GCN.

The idea is to classify each successive 15 frames individually and to predict whether they contains an action or not using a 1D-CNN. For every sub-sequences of 15 frames, we generate a label :

- 1 : Presence of action ;
- 0 : Absence of action.

We collect consecutive sub-sequences which contain actions. These collections of consecutive sub-sequences belong to one action. Then we inject them into a ST-GCN to classify the action as a fall or a non-fall. Our model is illustrated in figure 3.1.

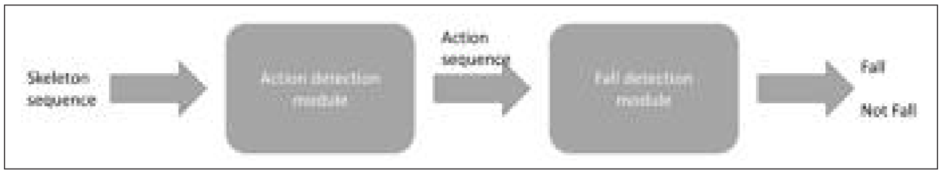


Figure 3.1 Real-time fall detection system

3.3.1 Action detection

Contrary to offline fall detection, online fall detection system receives a continuous stream of data containing different actions, hence the need for additional processing before classifying the action. Our action detection module is responsible for detecting the start and the end of each action.

3.3.1.1 Network architecture

Our model shown in figure 3.2 is composed of three convolutional layers followed by three fully connected layers and a softmax classifier. Each convolutional layer contains a 1D-CNN, a ReLu function and a 1D max pooling layer.

For each sub-sequence, we decide to use only four frames among 15 frames (frame id : 1,6,9,13) to reduce the complexity of the system and minimize the number of parameters. We also noticed that using only four frames slightly improves the result. The input of our model is a tensor of size (N, 4, 75) as shown in figure 3.2 where :

- N is the number of batches. Since we work in real time, the number of batches is one ;
- 4 is the number of frames and represents the number of channels ;
- 75 is the number of skeleton joint coordinates (there are 25 joints and each joint has 3 coordinates X,Y,Z).

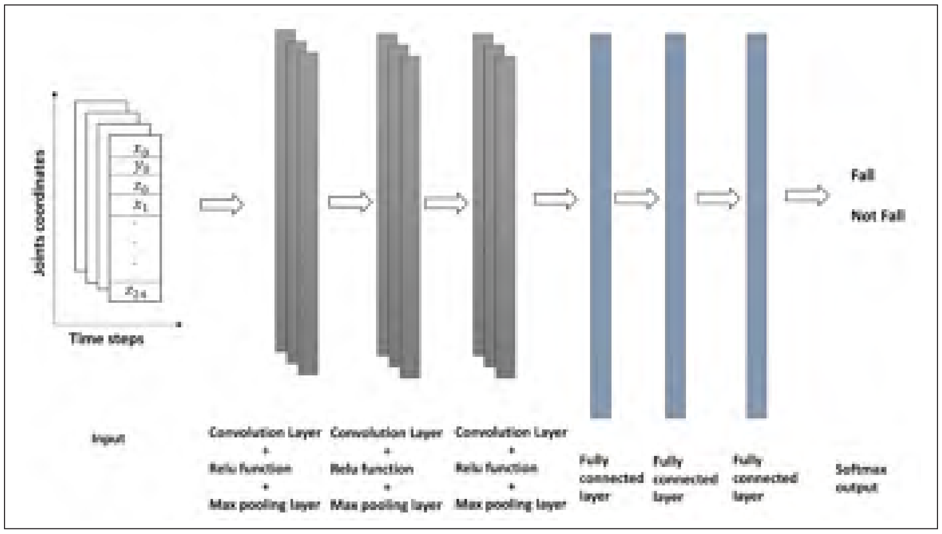

Joints coordinates
Time steps
Input
Convolution Layer + Relu function + Max pooling layer
Convolution Layer + Relu function + Max pooling layer
Convolution Layer + Relu function + Max pooling layer
Fully connected layer
Fully connected layer
Fully connected layer
Softmax output
Fall
Not Fall

Figure 3.2 Action detection module architecture

3.3.2 Fall detection

In (Oussema & Rita, 2020), we proposed a vision-based fall detection system based on a ST-GCN (Yu, Yin & Zhu, 2017). We decided to use this model because it achieved the best accuracy on the TSTv2 (Enea *et al.*, 2016) and Fallfree (Alzahrani *et al.*, 2017) datasets. This model takes as input a sequence of skeleton joints and performs a binary classification of each action. The output of classification is the class of the specific action : Fall or NotFall. This method has proved its generality and robustness and outperforms the previous state-of-the-art. It reaches 100% accuracy on TSTv2 (Enea *et al.*, 2016) and Fallfree (Alzahrani *et al.*, 2017).

3.4 Dataset

Our model is composed of two modules, an action detection module and a fall detection module. As presented in the last section, the fall detection module is already trained and validated using the TSTv2 (Enea *et al.*, 2016) and Fallfree (Alzahrani *et al.*, 2017) datasets.

Training the action detection algorithms requires a dataset of continuous videos which contains different actions. The PKU-MMD dataset is the perfect choice for our case. The PKU-MMD dataset was published in 2017. It is the biggest action detection and recognition oriented dataset. It was collected using three Kinect v2 cameras from three different view points. It contains 1076 videos. Each video lasts between three and four minutes and contains approximately 20 actions with 51 action classes. These actions were performed by 66 actors between 18 and 40 years old. This dataset is provided in five data modalities : RGB video, RGB images, Depth frames, Infrared (IR) sequences and skeleton joints. We used the PKU-MMD (Liu *et al.*, 2017) dataset to train the action detection module and evaluate the whole system.

3.4.1 Dataset and labelling

The label provided with PKU-MMD (Liu *et al.*, 2017) dataset is a file which contains the action ID and the start/end frame of each action. From these files we generate a new label that fits our model which requires a specific labeling. Two classes are considered :

- presence of action : some/all frames of the specific sub-sequence belong to an action, i.e., there is an overlap between the sub-sequence and start/end frame of any action.
- absence of action : all frames of the specific sub-sequence do not belong to an action

We decompose each video clip into sub-sequences of 15 frames. Using the labels of the dataset, we generate a new label for each sub-sequence. If the sub-sequence contains part of an action, we label it as 1, otherwise we label it as 0. An example is illustrated in figure 3.3.

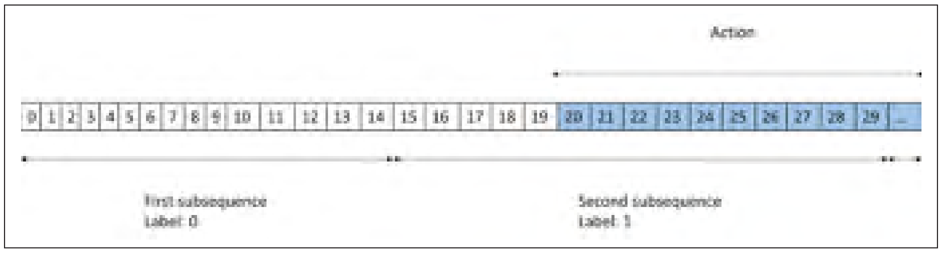


Figure 3.3 Labelling of PKU-MMD dataset

3.4.2 Evaluation

3.4.2.1 Evaluation metrics

The PKU-MMD dataset was developed to evaluate real-time action detection and recognition. Our objective is to detect only the fall events. For the evaluation step, we did not use the metrics recommended by the dataset authors since we applied some changes to the labels.

As we have a binary classification problem, we decide to use these metrics :

- Accuracy : represents the percentage of total samples classified correctly ;
- Sensitivity : represents the true positive rate ;
- Specificity : represents the true negative rate ;
- False Positive Rate.

3.4.2.2 Training and evaluation protocol

We decide to use the evaluation method suggested by the authors. We split the dataset into training and testing sets. So, we get 928 videos for training and 130 videos for testing. The subjects performing the actions in the training set are different from those in the testing set to prevent overfitting.

3.5 Experiments

We conduct two main experiments in order to test our method in real-time.

1. The objective of this experiment is to train our action detection module using the training set. Then we evaluate it using the testing set.
2. In this experiment, we test the whole fall detection system which contains the action detection and the fall detection modules using only the testing set.

3.5.1 Action detection

We conducted many experiments using the PKU-MMD dataset in order to chose the best configuration of the network. Some actors continue to move between actions and others stay motionless, and this made this dataset more challenging for our method to detect the start and end of an action. We decide to work with a sequence of successive 15 frames (duration of each sub-sequence = 0.5 seconds). For each sub-sequence, we pick four frames to be the input of our 1D-CNN. Our input size is 4x75, so we have designed a lightweight network. Our model was trained using stochastic gradient descent (SGD) optimizer and a learning rate lr=0.1.

3.5.2 Online fall detection system

In real-time time fall detection problem, two steps should be realised consecutively : extracting the action and classifying it. In this experiment, our objective is to evaluate the whole system which contains the action detection and fall detection modules and to figure out how these two modules interact.

We believe that PKU-MMD (Liu *et al.*, 2017) is the most appropriate dataset to test fall detection system in real time. It imitates real live actions ; we can see that the fall action happens at most once in the same video (one video contains around 20 ADL actions).

Our online fall detection system faces three main challenges in the evaluation step with this dataset :

- our fall detection module was trained using two fall detection-oriented datasets which contain one fall action and ten different ADL actions. We test it using a new action recognition oriented dataset which contains one fall action and 50 different ADL actions. So we have 40 more ADL actions. These additional actions could cause a increase of False Positive rate ;
- The PKU-MMD (Liu *et al.*, 2017) dataset is totally new to by the fall detection module. The difference of actors and view angles will affect the accuracy of the system ;
- our fall detection module was trained using a perfectly segmented action. In this experiment, actions may not be perfectly segmented using the action detection module. Errors in detecting the start and the end of the action will automatically affect the results of the action classification.

3.6 Results

3.6.1 Action detection

We reach an accuracy of 84.11%, a recall of 81.24%, a specificity of 87.07% and 6.37% FP rate on the test set. These metrics are good to represent the performance of the model. Although, this is not a classical binary classification. In this problem, each sample depends on the others and the distribution of the errors is important. We have actions that last for five sub-sequences each in average. A sub-sequence at the beginning of an action which is incorrectly classified will influence the results of classification of the whole action less than a classification error of a sub-sequence in the middle of the action. Similarly, a sub-sequence that does not belong to an action and is incorrectly classified will not have the same impact as a sub-sequence which belongs to an action and is incorrectly classified.

After analyzing the results, we notice that the action detection module detects almost all the actions but it reports nonexistent actions. In addition, we notice that in some cases the algorithm fails to classify a sub-sequence in the middle of the action. Therefore, two different actions are detected instead of one.

We illustrate three errors in figure 3.4 :

1. An error in classifying the last sub-sequence of the action (ELS) ;
2. An error in classifying a sub-sequence in the middle of the action (EMA) ;
3. An error in classifying three sub-sequences and report nonexistent action (RNA).

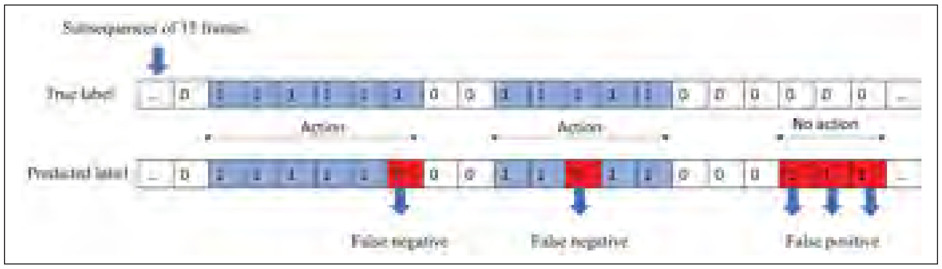


Figure 3.4 Action detection module classification errors

In ELS, the action detector correctly detects all the action's sub-sequences except the last one. In EMA, the action detector incorrectly classifies a sub-sequence in the middle of the action. Thus, two different actions are detected, but in reality this is a single action. We can notice that EMA is a more critical error than ELS.

In order to reduce the impact of EMA, we try to fix this error using an interpolation step. Our idea is to manually set the classification result of the current sub-sequence from 0 to 1 if this condition is met :

- the last sub-sequence is classified as '1'and the current sub-sequence is classified as 0 .

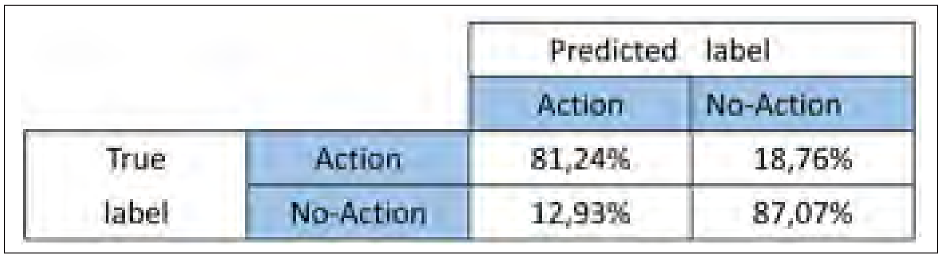
Using this step we significantly decrease the number of errors in the middle of the action. We report a recall of 89.04%, a specificity of 80.06% and 9.85% FP rate on the test set. The results are summarized in table 3.1.

We can notice the change of distribution of errors as shown in figure 3.5 and figure 3.6. We significantly increase the true positive rate from 81.24% to 89.04% and decrease the false

Table 3.1 Performance of the action detection module on PKU-MMD dataset. We report the results before and after applying the interpolation step

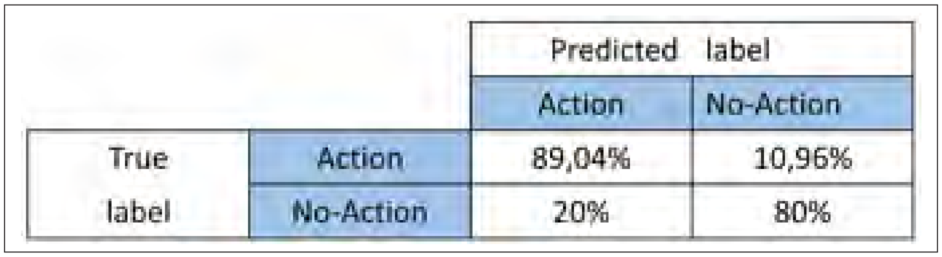
	Accuracy	**Recall/Sensitivity**	**Specificity**	**FP rate**
without interpolation	84.11%	81.24%	87.07%	6.37%
with interpolation	84.59%	89.04%	80.06%	9.85%

negative rate of from 18.76% to 10.96%. On the other hand, we decrease the true negative rate from 87.07% to 80% and increase the false positive rate from 12.93% to 20%. As we don't want to miss any sub-sequence that belongs to an action and raising false alerts is not a big problem, we prioritize true positives over the true negatives. Using the interpolation step, we overcome the EMA problem and consequently our module detects the whole action. Otherwise, we report a false negative rate of 10.96%. This could be due to an incorrect classification of the first or last subsequence of the action or to incorrect labelling in the dataset. This error could slightly influence the performance of the classification of this action.



		Predicted label	
		Action	No-Action
True label	Action	81,24%	18,76%
	No-Action	12,93%	87,07%

Figure 3.5 Confusion matrix before applying an interpolation step



		Predicted label	
		Action	No-Action
True label	Action	89,04%	10,96%
	No-Action	20%	80%

Figure 3.6 Confusion matrix after applying an interpolation step

In RNA, our module reports nonexistent action. After exploring the dataset, we notice that actors continue moving their feet between actions and this causes this kind of error. As we cannot fix this problem, we will rely on the fall detection module to classify foot motion as an ADL action.

3.6.2 Online fall detection system

As shown in table 3.2, we reach an accuracy of 90%, a recall of 100%, a specificity of 91% and 8.8% FP rate.

Table 3.2 Results of the online fall detection system

Accuracy	**Recall/Sensitivity**	**Specificity**	**FP rate**
90%	100%	91%	8.8%

When observing the results, we notice that most of the nonexistent actions are classified as ADL actions. We also notice that most of the false negatives are actions which are totally new to the fall detection module. The most important point is that all fall events are detected, which is our main objective.

Despite the fact that our fall detection module was trained using two fall detection datasets, it reaches very good results on a PKU-MMD (Liu *et al.*, 2017) dataset.

We test our method on a laptop equipped with a GPU (GTX 850m) ; it takes 4 s to analyze a sequence of four minutes which is considered fast enough for a real-time application.

3.7 Discussion

Our action detection module performs well in extracting actions from the video streams. These results are impressive but we noticed some flaws related to the action detection module :

- our detector always succeeds in detecting the presence of action but we notice some false positives. Fortunately, it does not present a problem for us because ST-GCN succeeds to classifing them as ADL actions ;

- we noticed that in some cases, the action detector does not detect the last action sequence, and in some cases in the middle of the action, a sequence is not classified as action. We could overcome this limitation by always classifying the last subsequent as action.

Results obtained in the whole fall detection system with the PKU-MMD (Liu *et al.*, 2017) dataset are very good. An interesting result is that our system produces excellent results on databases never seen before and contains around 40 new ADL actions. This demonstrates the generality of our fall detection module. Unfortunately, PKU-MMD does not contain occlusion scenes to test how our method faces real life scenarios containing occlusions.

3.8 Conclusions

The main objective of this study is to implement a model that detects falls in real-time using a Kinect v2 camera. We propose an online fall detection system using two modules, an action detection module based on a 1D-CNN and a fall detection module based on a ST-GCN. Our method exploits skeleton joints collected using a Kinect v2 camera. We use the PKU-MMD dataset to evaluate our fall detection system. Despite the fact that this is an unseen dataset and a challenging dataset for our fall detection module, we achieve very good results.

Through this work, we can see the limitations of the existing fall datasets. The authors of these datasets focused mainly on recording different types of fall (front fall, backward fall, lateral fall,...) and ignored the importance of the variety of ADL action. For example, there are only four different ADL actions in the TSTv2 dataset and less then ten different ADL actions in the Fallfree dataset. This lack of variety will limit the performance of the fall detection method when used in real life scenarios and when facing new actions. To overcome this limitation, we can fuse fall oriented dataset with action detection datasets which contains more than 50 different ADL actions.

Despite the promising results that we get, our system presents some imitations :

- we did not test our method in real life scenarios, but we believe that our method will give a very good result ;

- we did not test our method in occlusion scenarios, where the subject is partial or totally invisible.

Our action detection module can be applied with other offline action classifiers to provide a real-time system.

In our future work, we aim to reduce the false positive rate by using two synchronized Kinect V2 cameras and using both fall-oriented and action-oriented datasets to train the fall detection module. This would help solve the advantage to move beyond the problem of occlusion.

Acknowledgment

This paper and the research behind it would not have been possible without the support of Natural Sciences and Engineering Research Council, Aerosystems International Inc., and Prompt Quebec and Mitacs.

CONCLUSION ET RECOMMANDATIONS

Nous avons proposé dans ce mémoire un système fiable de détection de chute en temps réel. La majorité des travaux réalisés dans le domaine de détection de chute proposent des méthodes hors ligne qui nécessitent des traitements additionnels pour qu'ils puissent être appliqués en temps réel. Rares sont les méthodes qui sont proposées pour fonctionner en temps réel. Les algorithmes de détection de chute proposés dans la littérature sont généralement basés sur des méthodes de seuillage ou des algorithmes d'apprentissage machine classiques comme SVM et KNN. Ces algorithmes sont simples à implémenter et ne nécessitent pas trop de données. Néanmoins, une étape de sélection de caractéristiques manuelle est requise. Cette étape nécessite une bonne connaissance dans le domaine visé et influence d'une manière directe la performance du système. Les modèles d'apprentissage profond sont plus efficaces et fournissent de meilleurs résultats. Contrairement aux algorithmes classiques, les algorithmes d'apprentissage profond extraient automatiquement les caractéristiques les plus pertinentes. Cependant, ils nécessitent plus de données pour les entrainer, d'où le choix d'utiliser la technique de transfert d'apprentissage qui nous a permis de surmonter la limite d'insuffisance de données. Les entrées de notre modèle sont les coordonnées de squelettes fournis par la caméra Kinect v2. Nous avons utilisé quatre bases de données pour assurer la généralité et la robustesse du système proposé.

Notre système est composé de deux modules :

- un détecteur d'action basé sur le modèle 1D-CNN : Nous avons utilisé la base de données PKU-MMD pour entrainer et tester ce module. On a eu de très bons résultats ;
- un détecteur de chute basé sur le modèle ST-GCN : Nous avons utilisé la base de données NTU-RGB+D pour entrainer ce module. Puis on a effectué un transfert d'apprentissage en utilisant les bases de données TSTv2 et Fallfree. L'évaluation du module a été réalisée avec ces deux bases de données et on a atteint un excellent taux d'exactitude. Notre module a surpassé les autres méthodes de détection de chute dans la littérature.

Nous obtenons de très bons résultats dans le domaine de détection de chute principalement parce que :

- nous avons utilisé une caméra RGB-D et en particulier les données squelettiques qui sont indépendantes de l'environnement et des conditions d'éclairage ;
- nous avons utilisé l'algorithme d'apprentissage profond ST-GCN qui peut détecter la corrélation spatio-temporelle du graphe, dans notre cas, ce sont les données squelettiques ;
- nous avons utilisé l'algorithme d'apprentissage profond 1D-CNN pour détecter la présence d'actions, qui est à la fois léger et performant ;
- nous avons utilisé la technique de transfert d'apprentissage pour surmonter le problème d'insuffisance de données ;
- nous avons utilisé deux bases de données de chute durant la phase d'entrainement du module de détection de chute pour généraliser notre modèle.

On a pu détecter les chutes d'une manière efficace et fiable dont la performance a dépassé celle des méthodes proposées dans la littérature. Grâce à ce travail, nous pouvons voir les limites des bases de données de chutes existantes. Les auteurs de ces ensembles de données se sont principalement concentrés sur le signalement des différents types de chutes et négligent l'importance de la variété des actions de l'ADL. Pour surmonter cette limite, nous pouvons fusionner un ensemble de données orientées vers la détection de chute avec un ensemble de données de détection d'action qui contient plus de 50 actions ADL différentes. Malgré les résultats prometteurs, notre travail présente quelques limites :

- nous n'avons pas testé notre méthode dans des scénarios réels, mais nous croyons que notre méthode donnera de très bons résultats ;
- nous n'avons pas testé notre méthode dans des scénarios d'occlusion, où le sujet est partiellement ou totalement invisible ;
- notre module de détections de chute a été entrainé en utilisant deux bases de données de chutes. Ces deux bases de données ne couvrent que quelques actions ADL, ce qui a limité la

performance de notre système pour la classification de nouvelles actions. Pour surmonter cette limite, on peut fusionner ces deux bases de données avec une autre base de données qui contient beaucoup de classes d'actions comme NTU-RGB+D.

ANNEXE I

DOCUMENTATION AND REPRODUCIBILITY OF CODES

1. Getting Started

This project was developed using Python based on the Pytorch deep learning framework. The first step to replicate our results is to clone the project and create a virtual environment. Then, all the datasets used in this project will be downloaded and placed in a specific folder. The following steps are used to prepare the development environment :

1. Create a conda virtual environment
   > conda create -n fall-detection python=3.7 -y
2. Activate the environment
   > conda activate fall-detection
3. Install PyTorch and torchvision (CUDA is required)
   > conda install pytorch==1.2.0 torchvision==0.4.0 cudatoolkit=10.0 -c pytorch
4. Clone the project
   > git clone https ://github.com/ouss3ma/fall_detection
5. Install requirements
   > pip install -r requirements.txt
6. Download data
   The link to download the dataset is available in the github repository

2. Train model

The proposed model is composed of two modules, an action detection module and a fall detection module. Each module was trained separately using a specific dataset.

2.1 Train the offline fall detection module

The proposed model relies on a two-step process to train the fall detection module.

- train the ST-GCN network from scratch using the NTU-RGB+D dataset

  ```
  > python run.py /configs/recognition/st_gcn/xsub/train.yaml
  ```

  The train.yaml configuration file should have the following entries :

  ```
   name : '.processor.recognition.train'
   dataset_cfg :
  - name : '.datasets.SkeletonFeeder'
   data_path : ./data/NTU_RGB-D/xsub/train_data.npy
   label_path : ./data/NTU_RGB-D/xsub/train_label.pkl
  - name : '.datasets.SkeletonFeeder'
   data_path : ./data/NTU_RGB-D/xsub/val_data.npy
   label_path : ./data/NTU_RGB-D/xsub/val_label.pkl
  ```

  The following parameters were used during the training step and could be modified in the configuration file :

  - dropout layer with drop-rate of 0.5
  - stochastic gradient descent (SGD) optimizer with a learning rate of 0.01
  - Cross-Entropy Loss Function
  - batch size = 16
  - number of epochs = 100

  A validation step is conducted automatically every five epochs.

- perform a transfert learning based on the TST v2 and Fallfree datasets.

  ```
  > python run.py /configs/recognition/st_gcn/xsub/train.yaml
  ```

  The train.yaml configuration file should have the following entries :

```
name : '.processor.translearning.train'
dataset_cfg :
- name : '.datasets.SkeletonFeeder'
data_path : ./data/TST+Fallfree/xsub/train_data.npy
label_path : ./data/TST+Fallfree/xsub/train_label.pkl
- name : '.datasets.SkeletonFeeder'
data_path : ./data/TST+Fallfree/xsub/val_data.npy
label_path : ./data/TST+Fallfree/xsub/val_label.pkl
```

- evaluate the offline fall detection module

```
> python run.py configs/recognition/st_gcn/xsub/test.yaml
```

```
name : '.processor.translearning.test'
dataset_cfg :
- name : '.datasets.SkeletonFeeder'
data_path : ./data/TST+Fallfree/xsub/val_data.npy
label_path : ./data/TST+Fallfree/xsub/val_label.pkl
```

2.2 Train action detection module

- train the action detection module

  ```
  > python action_detection.py train
  ```

  The following parameters were used during the training step :

  - stochastic gradient descent (SGD) optimizer with a learning rate of 0.1
  - Cross-Entropy Loss Function
  - batch size = 256
  - number of epochs = 50

  A validation step is conducted automatically every five epochs.
- test the action detection module

  ```
  > python action_detection.py test
  ```

2.3 Evaluate the online fall detection system

```
> python onligne.py /data/PKU/
```

2.3.1 Results

The figure below is an example of the results achieved by our system.

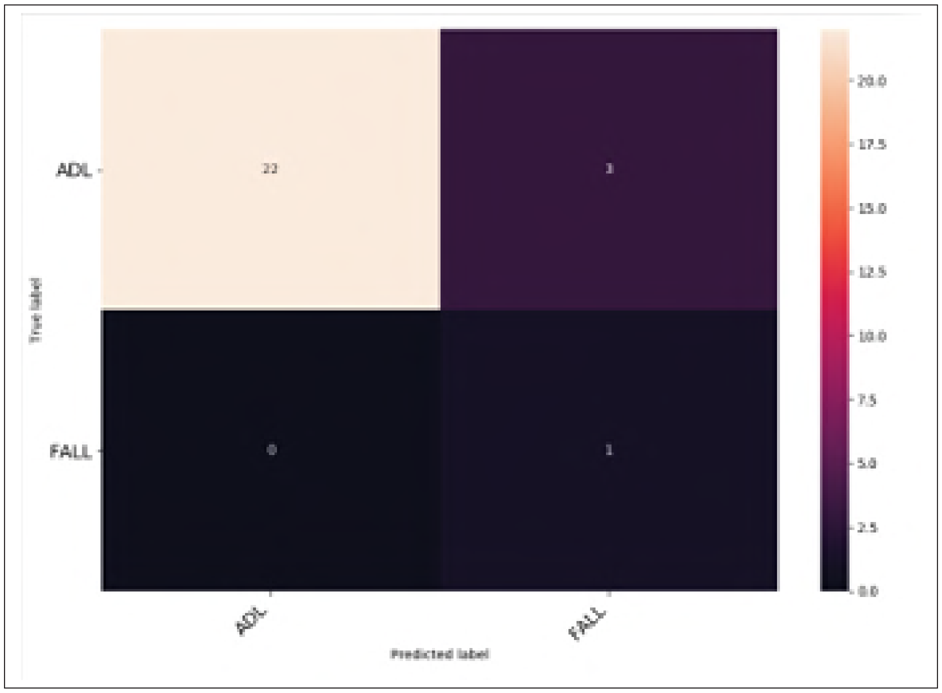


Figure-A I-1 Result exemple

BIBLIOGRAPHIE

Abedi, W. M. S. (2018). Unconsciousness Detection Supervision System Using Faster RCNN Architecture. *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, (ICFNDS '18). doi : 10.1145/3231053.3231094.

Alzahrani, M. S., Jarraya, S. K., Salamah, M. A. & Ben-Abdallah, H. (2017, Dec). FallFree : Multiple Fall Scenario Dataset of Cane Users for Monitoring Applications Using Kinect. *2017 13th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pp. 327-333. doi : 10.1109/SITIS.2017.61.

Alzahrani, M. S., Jarraya, S. K., Ali, M. S. & Ben-Abdallah, H. (2017). Watchful-Eye : a 3D skeleton-based system for fall detection of physically-disabled cane users. *International Conference on Wireless Mobile Communication and Healthcare*, pp. 107–116.

Alzahrani, M. S., Jarraya, S. K., Ben-Abdallah, H. & Ali, M. S. (2019). Comprehensive evaluation of skeleton features-based fall detection from Microsoft Kinect v2. *Signal, Image and Video Processing*, 13(7), 1431-1439. doi : 10.1007/s11760-019-01490-9.

Aziz, O., Musngi, M., Park, E. J., Mori, G. & Robinovitch, S. N. (2017). A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials. *Medical & biological engineering & computing*, 55(1), 45–55.

Cao, X., Kudo, W., Ito, C., Shuzo, M. & Maeda, E. (2019). Activity recognition using ST-GCN with 3D motion data. *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pp. 689–692.

Chen, Y., Li, W., Wang, L., Hu, J. & Ye, M. (2020). Vision-Based Fall Event Detection in Complex Background Using Attention Guided Bi-Directional LSTM. *IEEE Access*, 8, 161337–161348.

Chenguang, Y., Hu, J., Min, W., Deng, Z., Zou, S. & Min, W. (2020). A novel real-time fall detection method based on head segmentation and convolutional neural network. *Journal of Real-Time Image Processing*. doi : 10.1007/s11554-020-00982-z.

Chua, J.-L., Chang, Y. C. & Lim, W. K. (2015). A simple vision-based fall detection technique for indoor video surveillance. *Signal, Image and Video Processing*, 9(3), 623–633.

Cippitelli, E., Fioranelli, F., Gambi, E. & Spinsante, S. (2017). Radar and RGB-Depth Sensors for Fall Detection : A Review. *IEEE Sensors Journal*, 17(12), 3585-3604. doi : 10.1109/jsen.2017.2697077.

De Miguel, K., Brunete, A., Hernando, M. & Gambao, E. (2017). Home camera-based fall detection system for the elderly. *Sensors*, 17(12), 2864.

de Quadros, T., Lazzaretti, A. E. & Schneider, F. K. (2018). A movement decomposition and machine learning-based fall detection system using wrist wearable device. *IEEE Sensors Journal*, 18(12), 5082–5089.

Delahoz, Y. S. & Labrador, M. A. (2014). Survey on fall detection and fall prevention using wearable and external sensors. *Sensors*, 14(10), 19806–19842.

Di Giamberardino, P., Iacoviello, D., Tavares, J. M. R. & Jorge, R. N. (2012). *Computational Modelling of Objects Represented in Images III : Fundamentals, Methods and Applications*. CRC Press.

Dutta, A., Riba, P., Lladós, J. & Fornés, A. (2019). Hierarchical stochastic graphlet embedding for graph-based pattern recognition. *Neural Computing and Applications*, 1–18.

Enea, C., Ennio, G., Samuele, G. & Susanna, S. (2016). TST Fall detection dataset v2. IEEE Dataport. doi : 10.21227/H2QP48.

Fakhrulddin, A. H., Fei, X. & Li, H. (2017). Convolutional neural networks (CNN) based human fall detection on body sensor networks (BSN) sensor data. *2017 4th International Conference on Systems and Informatics (ICSAI)*, pp. 1461–1465.

Fan, K., Wang, P., Hu, Y. & Dou, B. (2017). Fall detection via human posture representation and support vector machine. *International journal of distributed sensor networks*, 13(5), 1550147717707418.

Gasparrini, S., Cippitelli, E., Gambi, E., Spinsante, S., Wåhslén, J., Orhan, I. & Lindh, T. (2015). Proposal and experimental evaluation of fall detection solution based on wearable and depth data fusion. *International Conference on ICT Innovations*, pp. 99–108.

Han, Q., Zhao, H., Min, W., Cui, H., Zhou, X., Zuo, K. & Liu, R. (2020). A Two-Stream Approach to Fall Detection With MobileVGG. *IEEE Access*, 8, 17556-17566.

Hwang, S., Ahn, D., Park, H. & Park, T. (2017). Maximizing accuracy of fall detection and alert systems based on 3D convolutional neural network. *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 343–344.

Jokanović, B. & Amin, M. (2017). Fall detection using deep learning in range-Doppler radars. *IEEE Transactions on Aerospace and Electronic Systems*, 54(1), 180-189.

Kawatsu, C., Li, J. & Chung, C.-J. (2013). Development of a fall detection system with Microsoft Kinect. Dans *Robot Intelligence Technology and Applications 2012* (pp. 623–630). Springer.

Kong, Y., Li, L., Zhang, K., Ni, Q. & Han, J. (2019a). Attention module-based spatial–temporal graph convolutional networks for skeleton-based action recognition. *Journal of Electronic Imaging*, 28(04). doi : 10.1117/1.Jei.28.4.043032.

Kong, Y., Li, L., Zhang, K., Ni, Q. & Han, J. (2019b). Attention module-based spatial–temporal graph convolutional networks for skeleton-based action recognition. *Journal of Electronic Imaging*, 28(4), 043032.

Kostopoulos, P., Nunes, T., Salvi, K., Deriaz, M. & Torrent, J. (2015, 01). Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement. 2. doi : 10.5220/0005179100300036.

Kouyoumdjian, F., Schuler, A., Matheson, F. I. & Hwang, S. W. (2016). Health status of prisoners in Canada : Narrative review. *Canadian Family Physician*, 62(3), 215–222.

Kwolek, B. & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer methods and programs in biomedicine*, 117(3), 489–501.

Le, T.-L., Morel, J. et al. (2014). An analysis on human fall detection using skeleton from Microsoft Kinect. *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, pp. 484–489.

Lee, C. K. & Lee, V. Y. (2013). Fall detection system based on kinect sensor using novel detection and posture recognition algorithm. *International Conference on Smart Homes and Health Telematics*, pp. 238–244.

Li, B., Li, X., Zhang, Z. & Wu, F. (2019). Spatio-temporal graph routing for skeleton-based action recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 8561–8568.

Li, W., Zhang, Z. & Liu, Z. (2010). Action recognition based on a bag of 3d points. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pp. 9–14.

Liu, C., Hu, Y., Li, Y., Song, S. & Liu, J. (2017). PKU-MMD : A large scale benchmark for skeleton-based human action understanding. *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*, pp. 1–8.

Liu, J., Shahroudy, A., Perez, M. L., Wang, G., Duan, L. Y. & Kot Chichung, A. (2019a). NTU RGB+D 120 : A Large-Scale Benchmark for 3D Human Activity Understanding. *IEEE Trans*

Pattern Anal Mach Intell. doi : 10.1109/TPAMI.2019.2916873.

Liu, J., Li, Y., Song, S., Xing, J., Lan, C. & Zeng, W. (2019b). Multi-Modality Multi-Task Recurrent Neural Network for Online Action Detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9), 2667-2682. doi : 10.1109/tcsvt.2018.2799968.

Maldonado-Mendez, C. & Hernandez-Mendez, S. (2019). Fall recognition system using feature selection and SVM : an empirical study. *2019 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pp. 187–192.

Maldonado-Mendez, C., Solis, A. L., Rios-Figueroa, H. V. & Marin-Hernandez, A. (2017). Human fallen pose detection by using feature selection and a generative model. *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pp. 1–6.

Martin, M. S., Dorken, S. K., Colman, I., McKenzie, K. & Simpson, A. I. (2014). The incidence and prediction of self-injury among sentenced prisoners. *The Canadian Journal of Psychiatry*, 59(5), 259-267.

Mastorakis, G. (2018). Unrepresentative video data : A review and evaluation. *arXiv preprint arXiv :1811.11815*.

McKendy, L., Biro, S. & Keown, L. A. (2019). Overdose Incidents in Federal Custody, 2012/2013-2016/2017 Research Report.

Min, W., Yao, L., Lin, Z. & Liu, L. (2018). Support vector machine approach to fall recognition based on simplified expression of human skeleton action and fast detection of start key frame using torso angle. *IET Computer Vision*, 12(8), 1133-1140. doi : 10.1049/iet-cvi.2018.5324.

Musci, M., De Martini, D., Blago, N., Facchinetti, T. & Piastra, M. (2018). Online fall detection using recurrent neural networks. *arXiv preprint arXiv :1804.04976*.

Núñez-Marcos, A., Azkune, G. & Arganda-Carreras, I. (2017). Vision-Based Fall Detection with Convolutional Neural Networks. *Wireless Communications and Mobile Computing*, 2017, 1-16. doi : 10.1155/2017/9474806.

Oussema, k. & Rita, n. (2020). Vision-Based Fall Detection using ST-GCN. *Manuscript submitted for publication in IEEE Access.*

Perry, A. E., Marandos, R., Coulton, S. & Johnson, M. (2010). Screening tools assessing risk of suicide and self-harm in adult offenders : a systematic review. *International journal of offender therapy and comparative criminology*, 54(5), 803-828.

Pham, N. P., Dao, H. V., Phung, H. N., Van Ta, H., Nguyen, N. H. & Hoang, T. T. (2018). Classification different types of fall for reducing false alarm using single accelerometer. *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*, pp. 316–321.

Pierleoni, P., Belli, A., Palma, L., Pellegrini, M., Pernini, L. & Valenti, S. (2015). A high reliability wearable device for elderly fall detection. *IEEE Sensors Journal*, 15(8), 4544–4553.

Putra, I., Brusey, J., Gaura, E. & Vesilo, R. (2018). An event-triggered machine learning approach for accelerometer-based fall detection. *Sensors*, 18(1), 20.

Ramachandran, A. & Karuppiah, A. (2020). A Survey on Recent Advances in Wearable Fall Detection Systems. *BioMed Research International*, 2020, 1-17. doi : 10.1155/2020/2167160.

Ren, L. & Peng, Y. (2019). Research of fall detection and fall prevention technologies : A systematic review. *IEEE Access*, 7, 77702-77722.

Seredin, O. S., Kopylov, A. V., Huang, S. C. & Rodionov, D. S. (2019). A Skeleton Features-Based Fall Detection Using Microsoft Kinect V2 with One Class-Classifier Outlier Removal. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W12, 189-195. doi : 10.5194/isprs-archives-XLII-2-W12-189-2019.

Shahroudy, A., Liu, J., Ng, T. & Wang, G. (2016). NTU RGB+D : A Large Scale Dataset for 3D Human Activity Analysis. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1010-1019. doi : 10.1109/CVPR.2016.115.

Shahzad, A. & Kim, K. (2018). FallDroid : An automated smart-phone-based fall detection system using multiple kernel learning. *IEEE Transactions on Industrial Informatics*, 15(1), 35-44.

Sharaf, A., Torki, M., Hussein, M. E. & El-Saban, M. (2015). Real-Time Multi-scale Action Detection from 3D Skeleton Data. *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 998-1005. doi : 10.1109/WACV.2015.138.

Shiba, K., Kaburagi, T. & Kurihara, Y. (2017). Fall detection utilizing frequency distribution trajectory by microwave Doppler sensor. *IEEE Sensors Journal*, 17(22), 7561-7568.

Smith, H. P., Kaminski, R. J., Power, J. & Slade, K. (2019). Self-harming behaviors in prison : a comparison of suicidal processes, self-injurious behaviors, and mixed events. *Criminal Justice Studies*, 32(3), 264–286.

Sousa, M., Goncalves, R. A., Cruz, A. R. & de Castro Rodrigues, A. (2019). Prison officers' attitudes towards self-harm in prisoners. *Int J Law Psychiatry*, 66, 101490.

doi : 10.1016/j.ijlp.2019.101490.

Su, B. Y., Ho, K., Rantz, M. J. & Skubic, M. (2014). Doppler radar fall activity detection using the wavelet transform. *IEEE Transactions on Biomedical Engineering*, 62(3), 865–875.

Sucerquia, A., López, J. D. & Vargas-Bonilla, J. F. (2017). SisFall : A fall and movement dataset. *Sensors*, 17(1), 198.

Suriani, N. S., Rashid, F. N. & Yunos, N. Y. (2018). Optimal accelerometer placement for fall detection of rehabilitation patients. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 10(2-5), 25-29.

Thomas, J., Leaf, M., Kazmierczak, S. & Stone, J. (2006). Self-injury in correctional settings :“Pathology” of prisons or of prisoners ? *Criminology Public Policy*, 5(1), 193-202.

Tian, Y., Lee, G.-H., He, H., Hsu, C.-Y. & Katabi, D. (2018). RF-based fall monitoring using convolutional neural networks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3), 1–24.

Triantafyllou, D., Krinidis, S., Ioannidis, D., Metaxa, I., Ziazios, C. & Tzovaras, D. (2016). A real-time fall detection system for maintenance activities in indoor environments. *IFAC-PapersOnLine*, 49(28), 286–290.

Tsai, T.-H. & Hsu, C.-W. (2019). Implementation of Fall Detection System Based on 3D Skeleton for Deep Learning Technique. *IEEE Access*, 7, 153049-153059. doi : 10.1109/access.2019.2947518.

Verma, K. K., Singh, B. M. & Dixit, A. (2019). A review of supervised and unsupervised machine learning techniques for suspicious behavior recognition in intelligent surveillance system. *International Journal of Information Technology*. doi : 10.1007/s41870-019-00364-0.

Wang, J., Zhang, Z., Li, B., Lee, S. & Sherratt, R. S. (2014). An enhanced fall detection system for elderly person monitoring using consumer home networks. *IEEE transactions on consumer electronics*, 60(1), 23–29.

Wang, P., Li, W., Ogunbona, P., Wan, J. & Escalera, S. (2018). RGB-D-based human motion recognition with deep learning : A survey. *Computer Vision and Image Understanding*, 171, 118-139. doi : 10.1016/j.cviu.2018.04.007.

Wu, J., Wang, K., Cheng, B., Li, R., Chen, C. & Zhou, T. (2019). Skeleton Based Fall Detection with Convolutional Neural Network. *2019 Chinese Control And Decision Conference (CCDC)*, pp. 5266-5271.

Xu, T. & Zhou, Y. (2018). Elders' fall detection based on biomechanical features using depth camera. *International Journal of Wavelets, Multiresolution and Information Processing*, 16(02). doi : 10.1142/s0219691318400052.

Xu, T., Zhou, Y. & Zhu, J. (2018). New Advances and Challenges of Fall Detection Systems : A Survey. *Applied Sciences*, 8(3). doi : 10.3390/app8030418.

Yacchirema, D., de Puga, J. S., Palau, C. & Esteve, M. (2018). Fall detection system for elderly people using IoT and big data. *Procedia computer science*, 130, 603-610.

Yan, S., Xiong, Y. & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv :1801.07455*.

Yang, R. L. . C. X. . T. Z. . W. Z. . Z. C. . J. Si-GCN : Structure-induced Graph Convolution Network for Skeleton-based Action Recognition. doi : 10.1109/IJCNN.2019.8851767.

Yao, L. Y., Yang, W. & Huang, W. (2019). An Improved Feature-Based Method for Fall Detection. *Tehnicki Vjesnik-Technical Gazette*, 26(5), 1363-1368. doi : 10.17559/Tv-20190411015902.

Yu, B., Yin, H. & Zhu, Z. (2017). Spatio-temporal graph convolutional networks : A deep learning framework for traffic forecasting. *arXiv preprint arXiv :1709.04875*.

Zhang, J., Li, W., Ogunbona, P. O., Wang, P. & Tang, C. (2016). RGB-D-based action recognition datasets : A survey. *Pattern Recognition*, 60, 86-105. doi : 10.1016/j.patcog.2016.05.019.

Zhang, J., Li, W. & Ogunbona, P. (2017). Transfer learning for cross-dataset recognition : a survey. *arXiv preprint arXiv :1705.04396*.

Zheng, W., Jing, P. & Xu, Q. (2019). Action Recognition Based on Spatial Temporal Graph Convolutional Networks. *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, pp. 1–5.