Università Politecnica delle Marche
Corso di Dottorato di Ricerca in Ingegneria dell'Informazione
Curriculum in Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

---

Human action recognition and mobility assessment in smart environments with RGB-D sensors

Ph.D. Dissertation of:
Enea Cippitelli

Advisor:
Prof. Ennio Gambi

Curriculum Supervisor:
Prof. Francesco Piazza

XV edition - new series

Università Politecnica delle Marche
Corso di Dottorato di Ricerca in Ingegneria dell'Informazione
Curriculum in Ingegneria Biomedica, Elettronica e delle Telecomunicazioni

Human action recognition and mobility assessment in smart environments with RGB-D sensors

Ph.D. Dissertation of:
Enea Cippitelli

Advisor:
Prof. Ennio Gambi

Curriculum Supervisor:
Prof. Francesco Piazza

XV edition - new series

Università Politecnica delle Marche
Corso di Dottorato di Ricerca in Ingegneria dell'Informazione
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

To my family, and Chiara.

Acknowledgments

I would like to thank Prof. Ennio Gambi and Dr. Susanna Spinsante for their guidance, insights, time and understanding during the last years. You have made my PhD experience active and challenging and your support has been crucial for my personal and professional growth.

I would like to express my gratitude to Prof. Thomas Lindh and his team for the ideas and suggestions received during the visiting period at KTH.

Thanks also to Prof. Francisco Florez-Revuelta for his excellent insights and suggestions not only during the three months spent at Kingston University, but also in the following collaborations.

Thanks to Prof. Sandro Fioretti and to the research group of Biomedical Engineering of Polytechnic University of Marche for the support received during the tests in the Movement Analysis Laboratory.

I am grateful to all the members of "Gruppo pranzo Univpm", in particular the actual and former members of Telecommunication Systems Team, and also to the PhD students of EMC^2 society at Kingston University. I had a great time with you during breaks, lunches and trips. A special thank to Samuele Gasparrini for the technical contribution in many research activities, advices, and friendship.

Finally, I would like to thank my parents, my brothers, and Chiara. Everything would have been more difficult without your constant encouragement.

Ancona, February 2017

Enea Cippitelli

Abstract

Population aging is a growing phenomenon in modern society and Active and Assisted Living tools can be investigated to have smart environments supporting elderly people to age at home. The adoption of enhanced living environments leads to a better quality of life for elderlies, because they can live in their preferred environment, and also for the society, increasing efficiency of used resources.

This work is focused on the development of algorithms and solutions for smart environments exploiting RGB and depth sensors. In particular, the addressed topics refer to mobility assessment of a subject and to human action recognition.

Regarding the first topic, the goal is to implement algorithms for the extraction of objective parameters that can support the assessment of mobility tests performed by healthcare staff. The first proposed algorithm regards the extraction of six joints on the sagittal plane using depth data provided by Kinect. This solution is validated with a marker-based stereometric system considering the accuracy in the estimation of torso and knee angles during the sit-to-stand phase. A second algorithm is proposed to simplify the test implementation in home environment and to allow the extraction of a greater number of parameters from the execution of the Timed Up and Go test. Such parameters are related to the time duration of the entire test execution as well as some phases (sit-to-stand, turn, sit), in addition to the extraction of indices related to the length of the steps and to the angular velocity of arm swing. Kinect sensor is combined with an accelerometer placed on the chest of the subject, which allows to accurately identify the time instants of the steps in the walking phase and the inclination angle of the torso during the sit-to-stand phase. The use of data coming from different sensors requires the development of a synchronization algorithm based on the compensation of transmission and exposure times of frames acquired by Kinect, which are estimated for both versions of the sensor and for the three available data streams: RGB, infrared, depth. The developed synchronization algorithm can be used also for other applications that benefit from the joint usage of RGB-Depth and inertial data, for example in the detection of falls. Algorithms for the identification of falls exploiting the same configuration of the Timed Up and Go test are therefore proposed.

Regarding the second topic addressed, the goal is to perform the classification of human actions that can be carried out in home environment. Two algorithms for human action recognition exploiting skeleton joints of Kinect are therefore proposed. The first one is based on Activity Feature Vectors (AFV), which contain the most important postures extracted from a sequence of frames. The second one is named Temporal Pyramid of Key Poses (TPKP), it considers the bag of key poses model and represents the structure of the action with a temporal pyramid. The algorithms have been evaluated on publicly available datasets, achieving results comparable with the state-of-the-art in the datasets CAD-60, KARD, MSR Action3D.

Sommario

L'invecchiamento della popolazione è un fenomeno in crescita nella società moderna e strumenti per l'Active ed Assisted Living possono essere studiati per avere ambienti intelligenti che supportano le persone anziane nell'invecchiamento all'interno dell'ambiente domestico. L'adozione di ambienti di vita avanzati porta ad una migliore qualità della vita per gli anziani, in quanto possono vivere nel loro ambiente prescelto, ed anche per la società, aumentando l'efficienza delle risorse utilizzate.

Questo lavoro è focalizzato sullo sviluppo di algoritmi e soluzioni per ambienti intelligenti sfruttando sensori RGB e di profondità. In particolare, gli argomenti affrontati fanno riferimento alla valutazione della mobilità di un soggetto e al riconoscimento di azioni umane.

Riguardo il primo tema, l'obiettivo è quello di implementare algoritmi per l'estrazione di parametri oggettivi che possano supportare la valutazione di test di mobilità svolta da personale sanitario. Il primo algoritmo proposto riguarda l'estrazione di sei joints sul piano sagittale utilizzando i dati di profondità forniti da Kinect. Questa soluzione viene validata con un sistema stereofotogrammetrico basato su marker considerando la precisione nella stima degli angoli di torso e ginocchio durante la fase di sit-to-stand. Un secondo algoritmo viene proposto per facilitare la realizzazione del test in ambiente domestico e per consentire l'estrazione di un maggior numero di parametri dall'esecuzione del test Timed Up and Go. Tali parametri sono relativi alla durata temporale dell'intera esecuzione del test e di alcune sue fasi (sit-to-stand, turn, sit), oltre all'estrazione di indici relativi alla lunghezza dei passi e alla velocità angolare delle braccia. Il sensore Kinect viene combinato con un accelerometro posto sul petto del soggetto, il quale permette di identificare con precisione gli istanti temporali dei passi nella fase di cammino e l'angolo di inclinazione del busto durante la fase di sit-to-stand. L'uso di dati provenienti da sensori diversi richiede lo sviluppo di un algoritmo di sincronizzazione basato sulla compensazione dei tempi di trasmissione e tempi di esposizione dei frame acquisiti da Kinect, i quali vengono stimati per entrambe le versioni del sensore e per le tre tipologie di segnale disponibili: RGB, infrarosso, profondità. L'algoritmo di sincronizzazione sviluppato può essere utilizzato anche per altre applicazioni che possono beneficiare dell'utilizzo congiunto di dati RGB-Depth ed inerziali, ad esempio nella rilevazione delle cadute. Vengono quindi proposti algoritmi di identificazione della caduta che sfruttano la stessa configurazione del Timed Up and Go test.

Per quanto riguarda il secondo argomento affrontato, l'obiettivo è quello di effettuare la classificazione di azioni che possono essere compiute dalla persona all'interno di un ambiente domestico. Vengono quindi proposti due algoritmi di riconoscimento attività che sfruttano i joints dello scheletro di Kinect. Il primo è basato su Activity Feature Vectors (AFV), i quali contengono le posture più importanti estratte da una sequenza di frame. Il secondo è denominato Temporal Pyramid of Key Poses (TPKP), considera il modello bag of key poses e rappresenta la struttura dell'azione con una piramide temporale. Gli algoritmi sono stati valutati su dataset pubblicamente disponibili, raggiungendo risultati confrontabili con lo stato dell'arte rispetto ai dataset CAD-60, KARD, MSR Action3D.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

1.1. Context

Population aging is defined as the increasing share of older people, aged 60 and over, in the population, and it is a growing phenomenon in modern society [1]. Globally, life expectancy at birth will rise from 70 years in 2010 – 2015 to 77 years in 2045 – 2050 and to 83 years in 2095 – 2100. The aging process is most advanced in high-income countries. The country where most of the aged people live is Japan, where 33% were aged 60 years or over in 2015. Italy and Germany are quite close (28%) and Finland is the third one with a percentage of 27% [2]. This demographic change has an impact also on the economic side, and to support an aging population has therefore become a priority for many governments [3].

A new paradigm in information technology is named Ambient Intelligence (AmI), and it is *"aimed at empowering people's capabilities by means of digital environments that are sensitive, adaptive, and responsive to human needs, habits, gestures, and emotions"* [4]. AmI tools can be adopted to improve quality of life in different environments, for example sensors can acquire data in the house, maybe related to the presence of people, adapt the environment according to the user preferences and also predict their needs and behavior. In addition to home environment, AmI has many other potential applications in office, transport and industry, and can be also applied in safety systems and e-health domain [5].

The application of information technology and AmI to the aging population brings to Active and Assisted Living (AAL), also known as Ambient Assisted Living, which is aimed to develop solutions to help elderly people to age at home. The objectives of AAL have been identified as [6]:

- extending the time people can live in their preferred environment by increasing their autonomy, self-confidence and mobility;

- supporting the preservation of health and functional capabilities of the elderly, promoting also a better and healthier lifestyle for individuals at risk;
- enhancing security, preventing social isolation and supporting the preservation of the multifunctional network around the individual;
- supporting families and caregivers;
- increasing the efficiency and productivity of used resources in the ageing societies.

European Union is fostering research on AAL through its main research and innovation programme: Horizon 2020 (H2020). During the first four years of H2020, EU has invested 2€ billion in the challenge *"Health, Demographic Change and Wellbeing"*, which is part of the third pillar of Horizon 2020 named Societal Challenges. In addition to this, EU finances also projects through the AAL Joint Programme, which is set to bring new ICT-based products, solutions and service concepts onto the market within two to three years of the end of the funding period. Some projects that received funding from this programme were able to develop products that are close to be released on the market. Mylife project [7] is addressed to elderly people with reduced cognitive function and aims to provide simple and intuitive services that are adapted to their individual needs and wishes. HELP project [8] has been funded by AAL programme to support people with Parkinson Disease. In particular, the idea is to have wearable and home devices to monitor health parameters and body activity, and to automatically release a controlled quantity of drugs. Other projects are in the field of assistive robotics, like Domeo [9], or aim to support people with mental disorders, such as Home4Dem [10].

Tools and algorithms that can be applied to AAL and smart environments are interesting and challenging not only from the commercial point of view, but also from the academic side. Researchers working on AAL domain are usually concentrated on algorithms or applications. Rashidi and Mihailidis [11] reviewed algorithms for Human Activity Recognition (HAR), also known as Human Action Recognition, Context Modeling, since AAL systems represent different types of context information, Anomaly Detection, which aims to find patterns in data that do not conform to the expected behavior, Location and Identity Identification, which allows to track and to monitor elderlies, and Planning, to schedule daily plans helping dementia patients to carry out their daily activities. On the other hand, applications may belong to different areas, such as Monitoring of health parameters or detection of dangerous situations, Wandering Prevention, providing alarms for unexpected actions or navigation assistance tools or Cognitive Orthotics tools, which can be used for example in case of medication management.

In the aforementioned scenario, the objectives of this thesis include the adoption of RGB-D (Depth) sensor, i.e. Microsoft Kinect, to address different tasks. The choice of a vision-based sensor as the main source of data is first of all motivated by the fact that, differently from wearable sensors, it is less obtrusive and does not rely on the user that needs to wear the device and to charge the battery. Comparing the Kinect device with other unobrusive sensors, such as environmental sensors, the former one allows to have more information about the context, and also to extract the human behaviour from his/her movements and not from the signal generated by the interaction with the environment.

The first objective of this thesis is to develop an application which aims to extract parameters for mobility assessment of a human subject. Kinect is adopted to extract the coordinates of skeleton joints and to calculate some useful indices from a side-view mobility test. In order to increase the reliability of the estimation, being aware of an higher obtrusiveness level of the system, an Inertial Measurement Unit (IMU) is considered jointly with the RGB-D sensor. Synchronization algorithms have been developed to correctly associate samples from different sources and mobility assessment and fall detection applications have been implemented. The second objective of the thesis is to propose algorithms which can be generally applied to address the HAR task. The input data are represented by skeleton coordinates captured by a RGB-D sensor, and organized in publicly available datasets.

1.2. Structure of the thesis

The thesis is organized in six chapters, which describe and detail the proposed approaches for human action recognition and mobility assessment using RGB-D sensors. The thesis has the following structure.

Chapter 2 reviews the state-of-the-art about the two main topics addressed: mobility assessment and human action recognition. Regarding the former one, the main tool used to evaluate the mobility, and often to infer the risk of falling, is the Timed Up and Go test. Different techniques proposed to instrument the test are presented, with a focus on video-based approaches. Regarding the second main topic, different technologies used for HAR are also reviewed, considering in more detail algorithms based on RGB-D data. The publicly available datasets that can be used to evaluate the performance of HAR algorithms and mobility assessment systems are listed.

Chapter 3 describes an algorithm to extract the coordinates of six skeleton joints from depth frames captured by Kinect during the sit-to-stand movement followed by some steps. The algorithm works in side view and the performance in the estimation of joint coordinates is compared with other markerless algorithms considering infrared markers as reference. A further validation process is performed for the sit-to-stand phase using a ground-truth represented by a stereometric system.

Chapter 4 details the instrumented Timed Up and Go with Kinect and a wearable IMU. The data fusion of multimodal data has allowed to extract different indices not only related to timing, but also to gait and posture. A synchronization algorithm based on the compensation of delay times affecting Kinect frames and acceleration samples has been developed to correctly associate data from different sources. The same setup can be used also for fall detection, and three different algorithms based on wearable and RGB-D data fusion are described. Simple and effective fall detection algorithms can discern among different types of falls and actions with an accuracy higher than 90% on a dataset of 264 sequences collected from 11 young subjects.

Chapter 5 presents algorithms for human action recognition based on 3D joint coordinates extracted from Kinect. Different features representation methods have been investigated, providing also experimental results on several RGB-D datasets. The first proposed algorithm is based on Activity Feature Vectors (AFV) containing the most representative postures of a sequence of frames. The second algorithm, based on Temporal Pyramid of Key Poses (TPKP), adopts a bag of key poses model and introduces a temporal pyramid to represent the structure of the action. The parameters required by the TPKP algorithm are optimized using evolutionary computation. The performance evaluation of AFV algorithm has been carried out on different RGB-D datasets, resulting in precision and recall higher than the state-of-the-art of about 10% on KARD dataset and comparable to it on CAD-60 dataset, where the figures are a precision of 93.9% and a recall of 93.5% considering the leave-one-actor-out cross-validation test. The TPKP algorithm has achieved results comparable to the state-of-the-art on MSR Action3D dataset, i.e. an accuracy of 95.14% on the cross-subject test.

Conclusions are drawn in Chapter 6 where, after clarifying the contribution of this work, some future research directions are identified and a list of publications is provided. Finally, Chapter A in appendix describes two software developed to collect RGB-D datasets using Kinect v1 and Kinect v2, which are now publicly available.

Chapter 2.

Related works

The aim of this chapter is to review and discuss the most relevant works in two fields that can be applied to Active and Assisted Living scenario: mobility assessment, especially the estimation of parameters to classify the risk of falling, and the recognition of human actions. An overview of the techniques that can be used to develop these solutions is provided, then the discussion is focused on solutions based on vision and, more in details, RGB-D sensors.

2.1. Mobility assessment

Due to the global increasing of age, the problem of falls in elderly people has to be considered in our society and many researchers started working on fall detection developing wearable or not-wearable systems [12]. A fall event, especially if experienced by an elderly, may lead to immediate physical injuries such as cuts, abrasions, and fractures of bones, but also to psychological consequences. People are afraid of falling again and they may reduce the level of their physical activity, leading also to functional decline and depression [13]. Due to these consequences, the necessity to estimate the mobility grade of the person, and thus the risk of falling, is as important as the detection of the fall.

Since the fall risk assessment is performed only by observations of the test by healthcare personnel, many research works in the last years have been developed to extract some important parameters from clinical test, to the aim to help clinicians in the assessment process. Tools for gait analysis may be based on wearable sensors or on not-wearable devices, which can be vision sensors (marker-based or markerless systems) or pressure sensors placed on the floor [14]. One of the most used test to evaluate the fall risk is the Get Up and Go test [15]. This test requires to the patient to stand up from a chair, walk for a short distance (about 3 meters), turn, walk back to the chair and sit down. The measure of the time required by the subject to perform the whole test, introduced in the Timed Up and Go (TUG) test [16], is already a measure of its movement ability and therefore of its risk of falling. In fact, if the subject takes more than 30 seconds to perform the test, she is classified as a subject with mobility problems. However, some parameters can be measured considering sensors monitoring the test, such as angles of different joints, step width, cadence, and many others [14]. Sprint et al. [17] reviewed the works which proposed to instrument the TUG, distinguishing among the technologies adopted by the different solutions.

2.1.1. Wearable sensing

A large number of studies proposed an instrumented version of the TUG using wearable sensors equipped with sensing and communication and/or storage capabilities. They usually integrate an Inertial Measurement Unit (IMU) capturing acceleration samples and a communication interface to send data to a server and/or a memory card to store data. Greene et al. [18] proposed to use two IMUs attached on the anterior of each shank of the subject. In addition to acceleration data, they evaluated also the time required to complete the test using a stopwatch. Temporal gait parameters are extracted from the identification of heel-strike and toe-off events, and are considered in addition to other temporal indices (*walk time*, *cadence*, *turn time*, ...). Angular velocity parameters were computed during some phases of the test and help in the definition of the statistical model to extract differences between subjects who experienced a fall and those who did not. A mean test accuracy of 76.8% was achieved for retrospectively estimating falls risk in a population of 349 elderly adults. A waist-mounted accelerometer has been considered in [19]. The authors proposed a model to evaluate the fall risk using three tests to extract useful parameters: TUG, Alternate-Step Test (AST), where the subject is asked to alternatively place each foot onto and off of a platform, and the sit-to-stand with 5 repetitions, consisting in the movement of standing up and sitting down with arms folded. Regarding the TUG, they extracted timings from different phases of the test and also some features related to acceleration data calculated within specific intervals. The fall risk evaluation has been carried out considering a linear least squares model built from a selected subset of features extracted. SankarPandi et al. [20] proposed the use of a wrist-mounted accelerometer during TUG execution to classify disability levels. A number of 40 features were extracted from acceleration data and then processed using a selection algorithm. With a population of 321 elderlies, the algorithm has been able to estimate the disability level using the optimal subset of features with

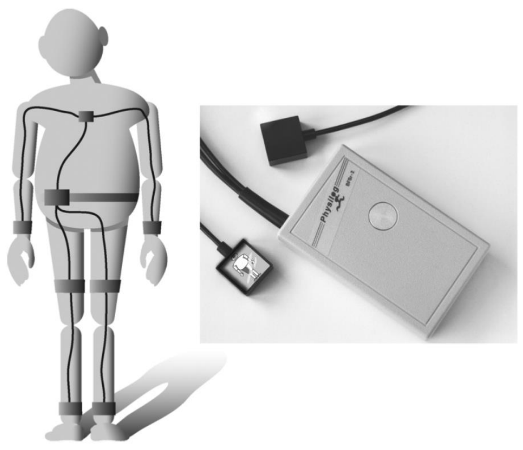


Figure 2.1.: Setup of the 7 wearable sensors adopted for the iTUG test (reprinted from [21]).

an accuracy of 62.16%, outperforming the classification achievable using only the time required to complete the test, which was 39.10%. Alternatives to the standard TUG, such as iTUG [21], have been also proposed. The authors exploited seven inertial sensors attached on the forearms, shanks, thighs and sternum, as shown in Figure 2.1, for the early detection of Parkinson's disease (PD). The walking path has been extended from 3 to 7 meters, extracting several parameters from sit-to-stand, gait, turning and turn-to-sit phases. For example, during the sit-to-stand phase, its temporal duration, the range of motion of the trunk, together with the average and peak angular velocity of trunk are considered. A 7 meters iTUG test has been considered also in [22]. A smartphone which integrates a tri-axial accelerometer has been placed on the lower back of the body and used to extract 28 parameters in the three sections of the iTUG: (sit-to-stand, gait, stand-to-sit). Dimensionality reduction is implemented considering Principal Component Analysis (PCA). A number of 10 principal components, which can be used for classification, were selected considering a dataset with 49 healthy adult subjects of different ages (in the range 28-87). Milosevic et al. [23] proposed a smartphone application (sTUG) for the evaluation of the iTUG. The application exploits accelerometers and gyroscope signals coming from the smartphone attached on the chest or belt. The extracted parameters are mainly related to the durations of the different phases of the test, even if angular velocities and maximum inclination of trunk angle during the sit-to-stand are also calculated. The Android application provides an immediate feedback to the user showing the calculated indices and allows to upload the results into the medical record. Silva and Sousa [24] presented a smartphone-based iTUG where the test has been segmented in three phases (stand up, walk forward and turn around) and the features extracted from each segment are statistical measures and frequency-based metrics obtained from the accelerometer magnitude signal. Considering a population of 18 older adults, the authors found that the walking and turning components were the most significant ones to differentiate between a higher risk and a lower risk person. Another wearable system, which is more flexible because it does not require a fixed test distance, has been proposed in [25]. The test can be executed from an iPhone or iPad and exploits data from 5 sensors placed on foot, shin and thigh of the patient, and equipped with triaxial accelerometer, gyroscope and magnetometer, and being able to provide quaternion representing orientation. The algorithms can calculate angles of knee and ankle which are validated with a marker-based VICON system. Gait parameters, such as stride length, the total distance traveled and average velocity are also extracted, together with the total time of the test.

2.1.2. Unobtrusive sensing

Researchers proposed also to instrument the TUG using unobtrusive technologies. Some research works considered the use of sensors placed on the floor monitoring the parameters related to the gait. Demura et al. [26] proposed also to place obstacles along the path and to evaluate different timing parameters in a modified TUG called TUGO (TUG test with an obstacle), such as the total time, going time or turn time, in addition to the single support time before and after the obstacle. In other cases, the floor sensor is adopted to compare the performance of other systems, such as video-based solutions. Baldewijns et al. [27] validated their method to extract gait information using Kinect depth data with GAITRite walkway. Results on 19 tests showed an average error of 2.48 cm for the step length and 0.24 seconds for the step time.

Other researchers working on video processing proposed solutions to extract objective parameters from TUG. In [28], two webcams have been used to capture the side and back view of the subject performing the test. Three categories of parameters are calculated, including: time of the walk phase, number of steps, stability into and out of the turn. The subject's silhouette has to be extracted from the video using background subtraction techniques and filtering methods to remove some artifacts. Then, most of the walking indices are estimated from the tracking of the centroid of head silhouette, while the steps are counted looking at the acceleration of the leading foot. Two calibrated cameras were also used in [29], where a 3D human

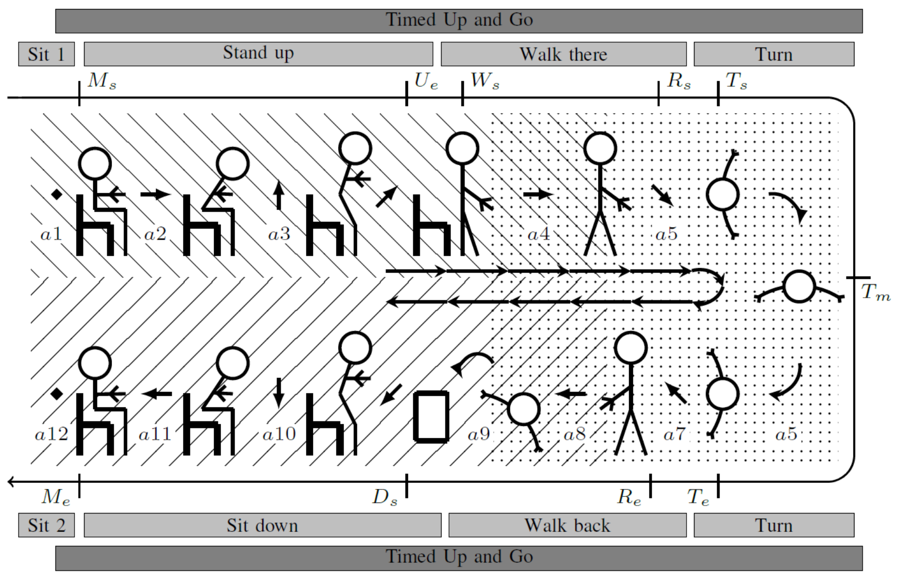


Figure 2.2.: Setup of the sTUG test, with different detected events during the test: M_s-M_e, and actions evaluated during assessment: a_1-a_{12} (reprinted from [30]).

model is reconstructed to evaluate the 180° turning movement. The turning time and the number of turning steps are extracted from 7 people aged between 25 and 88. The authors found that their method is able to correctly estimate the turning time, with a good match if compared to the expert rating.

With the availability of RGB-D devices, researchers started working also with these sensors. Lohmann et al. [30] proposed an approach called Skeleton Timed Up and Go (sTUG) using 2 kinect sensors: one records the front of the action area (dashed area in Figure 2.2) and the other one records the dotted area in Figure 2.2. A number of 9 events are extracted from the trajectories of skeleton joints given by the two calibrated sensors and durations of different phases are computed. Good results have been obtained from tests with 13 people comparing the time durations with a stopwatch. Kitsunezaki et al. [31] proposed to measure the time required to perform TUG with a Kinect placed at a distance of 4 meters from the chair where the subject starts the test. Considering 6 people and 3 executions for each person they found an average difference of 0.33 seconds from the value measured by the examiner. The same authors developed also a timed 10-meter walk test with two synchronized Kinect and a software to detect timing parameters where a person crosses a line. In this test, the same number of executions gave a difference of 0.15

seconds between the two sets of results (examiner and Kinect). Hassani et al. [32] proposed to extract some parameters from skeleton joint positions. In particular, the author were focused on two sub-parts of the TUG: the forward-upward (STS) and the downward-backward (BTS) displacements of the shoulder in the sagittal plane. Within these movements, they considered: movement duration, shoulder path curvature, trunk angle and ratio between vertical phase duration and horizontal phase duration. Then, the total duration of the TUG is computed as the time interval between the start of the forward phase and the end of the backward phase. Results on 10 healthy young subjects showed that, in 70% of the cases, the mean value of ratio, trunk angle and duration are within their confidence intervals, which means that the system provides reliable measurements and the variability is reduced among different subjects. The same authors improved the method by estimating sitting posture with a linear Support Vector Machine (SVM) [33]. Considering two groups of people, young and elderly, the authors noticed that the trunk angle in young group was more than twice angle in elderly group: $16.76°$ and $35.81°$ during STS movement, and this parameter can show the effect of aging. Recently, Gianaria et al. [34] used Microsoft Kinect v2 to extract some parameters form TUG test: TUG test total time, walking time (τ), covered distance, walking speed (β), swing time, double support time, torso inclination angle. Some tests performed on 30 elderly and 6 young people provided meaningful results. For example, walking-related parameters were $\beta = 0.75$ m/s and $\tau = 9.31$ s for seniors, $\beta = 0.92$ m/s and $\tau = 7.16$ s for youth.

2.1.3. RGB-D Datasets for mobility assessment

In order to foster the research in this field, Leightley et al. [35] recorded a dataset with health-related actions, called Kinect 3D Active (K3Da). The dataset includes motion collected from young and elderly people, aged between 18 and 81. With data captured from 54 people (32 men and 22 women) this is the largest RGB-D dataset containing clinically supported motion sequences, ensuring also high inter-individual variability in age and physical capabilities. The RGB-D device used is the Kinect v2, skeleton data (25 joints at 30 fps) and depth frames (512×424 at 30 fps) are the provided data. Timed Up and Go is included in the tests provided in the K3Da dataset, and samples are collected with the sensor fixed horizontally to a tripod at a height of 0.7 m in a controlled environment, without room furniture and standardised room lighting.

2.2. Human action recognition

Human action recognition is a research topic which has been investigated a lot in the last years. The main reason is that HAR may enable different applications, from human computer interaction to video surveillance, including also assistive applications, such as the monitoring of people living alone. HAR in AAL scenario can be implemented considering different technologies, which may involve wearable inertial sensors, ambient/environmental sensors, acoustic sensors, radio-based techniques or vision-based devices.

2.2.1. Wearable sensing

Wearable sensors can be placed at different positions and the generated signals, which are related to the movements, can be used to distinguish among a set of activities. Data from wearable devices for HAR can be handled with two main approaches. Data can be captured by a smartphone, and the same device can be also exploited for classification purpose [36]. A more general architecture involves sensors integrated on small devices equipped with communication interfaces and able to send data to an integration device, represented by a smartphone or a laptop, that can store and process data or send them to a remote server [37]. Considering only inertial data, many different features have been proposed, and the inclusion of the features in the classification model has to be evaluated to reduce the complexity of the model and to increase the classification accuracy due to a better separation among the classes [38]. Wearable sensors are quite cheap and, if compared for example to cameras, they produce a lower amount of data, enabling a processing step with less computational resources. On the other hand, one of the main issues about the use of wearable sensors is their placement [39]. The sensors are usually placed on the sternum, lower back, and waist. Waist-placement is closer to the center of mass of the human body and can better represent most human motions. However, the placement should take into account also the movements involved in the set of activities that has to be recognized. In fact, if the use of a waist-placed accelerometer can efficiently classify activities and resting positions involving the whole body, such as walking, sitting, lying and also falls [40], the inclusion in the dataset of activities including upper body movements, such as brushing teeth and working at computer, requires a different placement of the sensor, for example on the wrist [41]. Finally, the classification performance of an activity recognition algorithm can be increased considering more sensors placed on the human body [42], increasing also the level of obtrusiveness.

2.2.2. Unobtrusive sensing

Unobtrusive environmental sensors may be exploited to recognize human activities. However, since these techniques are based on data acquired by sensors placed on objects, the activities are not directly related to the human movement, but they are mostly related to the interaction of the human with the object or the environment. The human behaviour can be inferred from the object interaction [43]. State-changes sensors may be installed on doors, windows, cabinets, and also on some objects such as owens, fridges or washing machines, and can therefore reveal activities such as preparing lunch, toileting or doing laundry [44]. In addition to state-changes sensors, passive infrared sensors (PIRs) can be used to detect the presence of people within an environment, and the fusion of the data can increase the level of confidence in the activity estimation process. Ordóñez et al. [45] proposed different machine learning algorithms to classify datasets made of 10 activities using three categories of sensors: PIR sensors to detect people presence, state-changes sensors to detect opening and closing activities of doors and cupboards and float sensors to detect the flushing of toilet. Break-beam sensors to detect motion and chair-mounted pressure mats to detect static occupants can also be employed in home environment [46]. Despite the big advantages of being completely unobtrusive and privacy preserving, HAR systems based on environmental sensors require a not negligible time to be installed, a limited amount of information can be inferred from the sensors, and do not detect dangerous situations, such as falls.

Another unobtrusive technology adopted for human activity recognition in home environment is based on microphones to detect sounds. The idea is similar to ambient sensors, and the events are detected considering the sound generated by the interaction between people and objects or the environment. Some activities, such as chatting or reading newspapers can be easily detected by their sounds, and different activities at a dining table can be categorized using only a microphone, recognizing also the type of food or drink [47]. Considering more microphones placed in different positions of the apartment, some activities such as vacuuming, blender mixing or events as oven alarm or door slam can be detected [48]. Multiple acoustic sensors may be set up as a Wireless Sensor Network (WSN), considering that the main challenges can be the limited processing power and working memory for the end devices. However, for indoor context, feature extraction schemes with low complexity may be equally effective as the high-cost ones [49]. Clinically relevant activities may be also detected considering an acoustic WSN to efficiently monitor elderly people. Vuegen et al. [50, 51] proposed a network of 7 nodes to be placed in 3 rooms to detect 10 different activities. Being particular ambient sensors, the use of acoustic sensors has the advantage of being privacy preserving, and, even if they may be subject to noise, they can be used to detect also falls [52]. Considering an array of microphones, it is possible to consider only sounds located below a certain height from the floor, reducing the false alarm rate and estimating also the 3D sound location [53].

An alternative activity recognition method is based on radio techniques, which take advantage of body attenuation or characteristics of channel fading to recognize human activities or gestures [54]. Even if no physical sensing module is required, the user may be asked to wear a wireless transceiver implementing ZigBee, WiFi or RFID standards. Some methods, on the other hand, can work without any device on the human body, exploiting for example the WiFi links established between smart appliances and the access point [55] or FM signal strength which is correlated with the receiver's position [56]. Micro-Doppler radar signatures represent another solution and, through the usage of commercial radar motes, a small set of activities, such as walking, running, and crawling, can be discriminated with a high classification rate [57].

Vision-based sensors have been extensively used for human activity recognition. Vision sensors can be adopted in a outdoor environment, for surveillance applications, as well as in a indoor scenario, where they can provide more information about the environment with respect to other sensors. Many different reviews on vision-based HAR techniques have been published in the past, each of which proposing its own taxonomy and focusing on different aspects. At the beginning of the recognition of human motion, Aggarwal and Cai [58] provided a review covering three areas: motion analysis based on human body parts, tracking human motion (without using the body parts) from a single view or multiple perspectives, recognizing human activities from image sequences. Wang et al. [59] identified three major issues in the process of human motion analysis: human detection, mainly constituted by motion segmentation and object classification, human tracking and human behavior understanding. Turaga et al. [60] proposed to distinguish between actions and activities. They labelled as actions the movements characterized by simple motion patterns, usually performed by one person, such as bending, walking, swimming. Activities are more complex and involve more people who could be interacting with each other in a constrained manner, such as two persons shaking hands or a football team scoring a goal. However, different definitions of actions and activities have been proposed by others. Chaaraoui et al. [61] discern between action and activity considering the degree of semantics ad the amount of time required to perform the analysis. Considering this approach, actions are movements with a duration of seconds, such as sitting, standing or walking. Activities are made up a set of actions in a time frame from tens of seconds to units of minutes. Example of activities can be cooking, taking a shower or making the bed. Even if HAR in the context of AAL is often related to Human Activity Recognition, in this thesis, considering the discrimination between actions and activities proposed in [61], the work is mostly concentrated on shortest movements and with a low level of semantics, thus the acronym HAR can be expressed as Human Action Recognition.

The recent release of low cost depth sensors fostered the recognition of human motion using 3D data. Depth information helps to overcome some issues in human action recognition based on RGB images, such as the presence of shadows, light reflections, similarity of colors between foreground and background which may affect people silhouette segmentation [62]. Aggarwal and Xia [63] divided the methods to obtain 3D data into three different categories: marker-based motion capture systems, stereo images or range sensors. In addition to the technology adopted to obtain depth data, they organized the reviewed methods into five categories considering the features: features from 3D silhouettes, features from skeletal joint or body part locations, local spatio-temporal features, local occupancy patterns, and 3D scene flow features.

In this work, the review of previously published methods based on RGB-D sensors is organized considering the data exploited by the HAR algorithm. Some methods may exploit only depth or skeleton data, others can be based on the fusion of different input data, combining for example depth frames or RGB images with skeleton joints. Some methods were aimed to extract features from depth data, such as [64], where the main idea is to evaluate spatio-temporal depth subvolume descriptors. A group of hypersurface normals (polynormal), containing geometry and local motion information, is extracted from depth sequences. The polynormals are then aggregated to constitute the final representation of the depth map, called Super Normal Vector (SNV). This representation can include also skeleton joint trajectories, improving the recognition results when people move a lot in a sequence of depth frames. Depth images can be seen as sequence features modeled temporally as subspaces lying on the Grassmann manifold [65]. This representation, starting from the orientation of the normal vector at every surface point, describes the geometric appearance and the dynamic of human body without using joint position. Other works proposed holistic

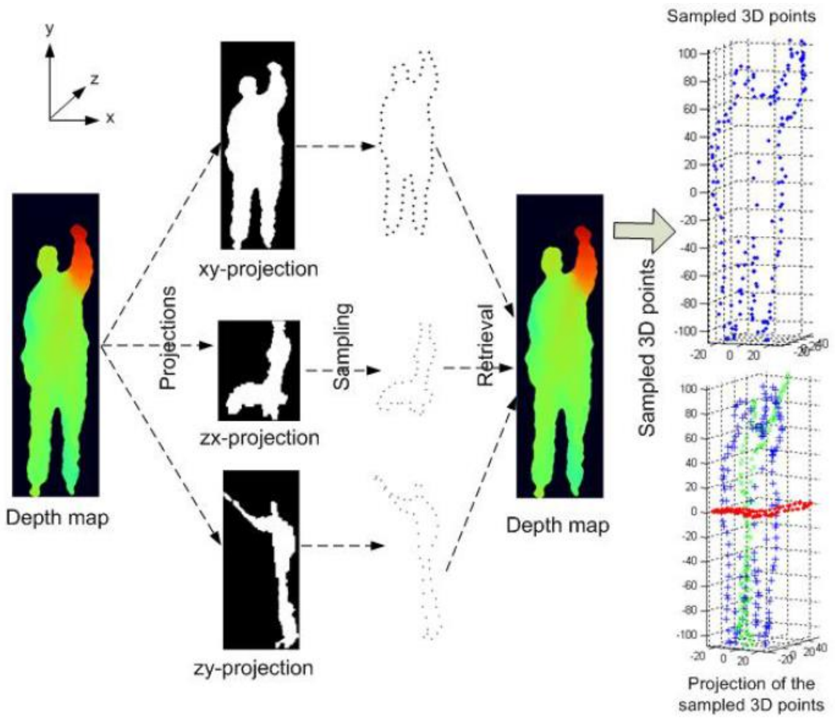


Figure 2.3.: Sampling of 3D points using the silhouette-based method for HAR (reprinted from [68]).

descriptors: the HON4D descriptor [66], which is based on the orientations of normal surfaces in 4D, and Histogram of Oriented Principal Components (HOPC) descriptor [67], which is able to represent the geometric characteristics of a sequence of 3D points. HOPC descriptor can be also used in a local setting, where local HOPC are extracted at candidate spatio-temporal keypoints (STKP). A HOPC quality factor is defined to rank the STKPs and to discard the low quality ones. Depth data make easier the process of silhouette extraction and some algorithms for action recognition exploiting 3D silhouettes have been proposed. Li et al. [68] developed a method that represents postures considering a bag of 3D points extracted from depth data, shown in Figure 2.3. Only a small set of 3D points is considered, and a projection method has been developed to sample the representative 3D points by performing planar projections of the 3D depth map and extracting the points that are on the contours. The temporal relationship among the postures is modeled using an action graph, where each node represents a salient posture. Chaaraoui et al. [69] proposed to use the points belonging to the contour of the human silhouette, and to extract

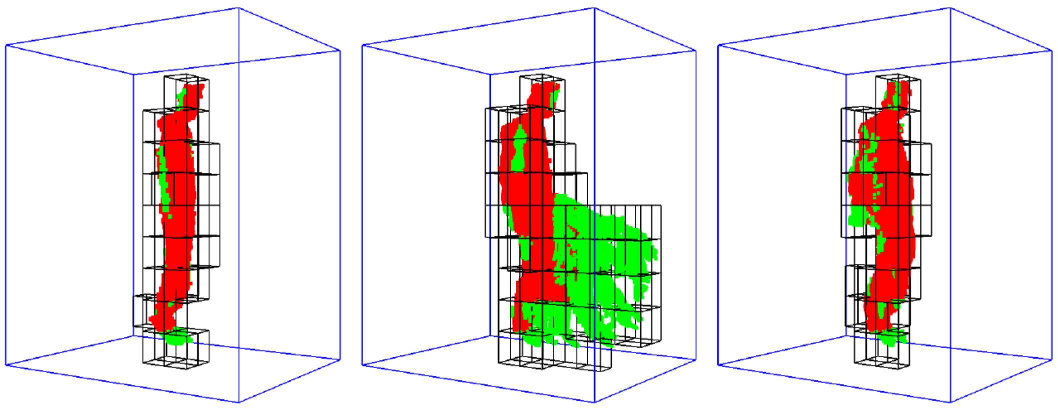


Figure 2.4.: Spatio-temporal cells representation of the action *forward kick* (reprinted from [71]).

features from them. The learning procedure consists in the extraction of the *key poses* and the action, represented as a set of feature frames, is then constituted by known characteristic poses. Other interesting features are represented by local Spatio Temporal Interest Points (STIPs) applied to depth data [70]. Depth-based STIPs include a noise suppression scheme which can handle some characteristics of the depth images, such as the noise in the borders of an object, where the depth values show a big difference in the transition from foreground to background, or the noise given by errors in the depth estimation algorithm, which can result in some gaps in the depth map. In [70], the use of depth STIPs is combined with a descriptor containing the spatio-temporally windowed pixels within a 3D cuboid centered at the interest point. A codebook is built by clustering the identified cuboids and an action can be represented as a sequence of elements from the codebook. Vieira et al. [71] proposed to divide the spatio-temporal space into multiple segments. A 4D grid is obtained, where a saturation scheme is employed to enhance the role of the cells corresponding to the moving parts of the body. Figure 2.4 shows the spatio-temporal cells for the action *forward kick*, which is divided in three temporal segments, and all the frames are placed together in the same space. Red points are those in the cells with more than a fixed amount of points. The occupancy patterns are then extracted and Principal Component Analysis (PCA) is used as a dimensionality reduction method.

Other works exploit both depth and skeleton data, for example the *3.5D representation* combines the skeleton joint information with features extracted from depth images, in the region surrounding each node of interest [72]. The features

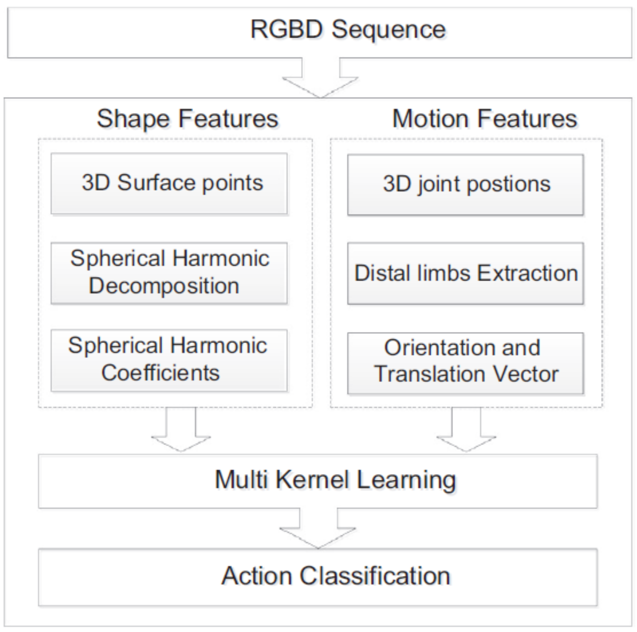


Figure 2.5.: Method for human action recognition using multiple features and MKL technique (reprinted from [73]).

are extracted using an extended Independent Subspace Analysis (ISA) algorithm by applying it only to local region of joints instead of the entire video, thus improving the training efficiency. Depth and skeleton features can be combined at different levels of the activity recognition algorithm. Althloothi et al. [73] proposed a method where the data are fused at the kernel level, instead of the feature level, using the Multiple Kernel Learning (MKL) technique which combines and utilizes multiple kernels in the process of learning kernel based classifiers. As shown in Figure 2.5, the 3D silhouette structure is described using shape features, extracted from the depth map using spherical harmonics representation. The motion features are represented by distal limb segments extracted from the joint positions. The initial frame is taken as a reference and each distal limb segment is described by the orientation and translation distance with respect to it. On the other hand, fusion at the feature level of spatiotemporal features from depth data and skeleton joints is performed in [74]. The spatiotemporal features represent local motions at different 3D interest points, and the skeleton joints features represent spatial locations of body parts. In such a work, several spatiotemporal interest point detectors, such as Harris 3D [75], ESURF [76], HOG3D [77] have been fused using regression forests with the skele-

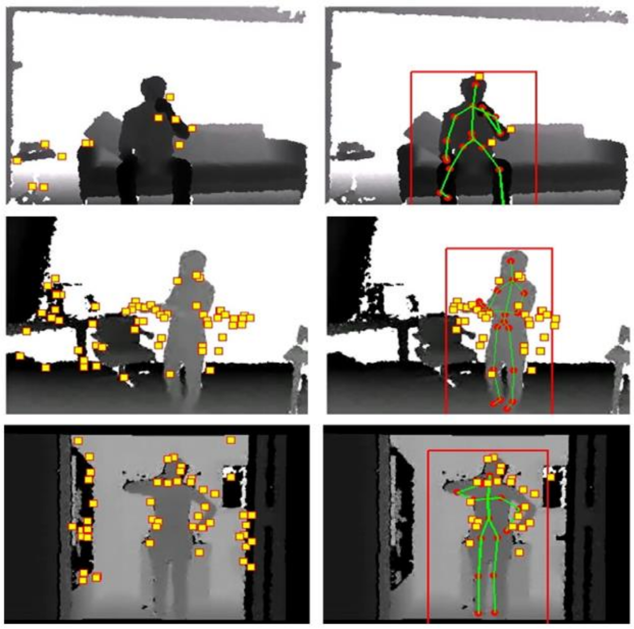


Figure 2.6.: Results of STIPs refinement using a bounding box extracted from skeleton joints (reprinted from [80]).

ton joint features consisting of posture, movement and offset information. Features extracted from human silhouette can be concatenated with normalized skeleton features, to improve the recognition rate performing a feature-level fusion [78]. The simple feature concatenation as a fusion technique has been allowed by the very low-dimensional features: the resulting increased size of the final feature is not critical. The feature concatenation has been used to keep all the characteristic information provided by both feature types.

Skeleton joints extracted from depth frames can be combined also with RGB data. Luo et al. [79] proposed a human action recognition framework where the pairwise relative positions of joints and Center-Symmetric Motion Local Ternary Pattern (CS-Mltp) features from RGB are fused both at feature level and at classifier level. STIPs are typically used in activity recognition where data are represented by RGB frames. This approach can be also extended to depth and skeleton data, combining the features with Random Forests [80]. Different combinations of detectors and descriptors are considered, and the combination of depth STIP features with RGB videos or skeleton joint positions may help to have a better understanding of depth-based features, enhancing the recognition performance, as shown in

Figure 2.6. Instead of using spatio-temporal features, another approach for human activity recognition relies on graph-based methods for sequential modeling of RGB data. This concept can be extended to depth information, and an approach based on coupled Hidden Conditional Random Fields (cHCRF) model, where visual feature sequences are extracted from RGB and depth data, has been proposed [81]. The main advantage of this approach is the capability to preserve the dynamics of individual sequences, even if the complementary information from RGB and depth are shared.

Other works simply rely on Kinect skeleton data because they represent a compact and effective description of the human body. In many cases they can achieve performance very close to the algorithms exploiting multimodal data, and sometimes they also perform better than those solutions. Devanne et al. [82] proposed to represent human actions by spatio-temporal motion trajectories in a 60-dimensional space since they considered 20 joints, each of them with 3 coordinates. Then, an elastic metric, which means a metric invariant to speed and time of the action, within a Riemannian shape space, is employed to represent the distance between two curves. Finally, the action recognition problem can be seen as a classification in the Riemannian space, using a k-Nearest-Neighbor (k-NN) classifier. Other skeleton representations have been proposed. The APJ3D representation [83] is constituted starting by a subset of 15 skeleton joints, from which the relative positions and local spherical angles are computed. After a selection of key-postures using k-means clustering algorithm, the action is partitioned using a reviewed Fourier Temporal Pyramid [84] and the classification is made by Random Forests. Another joint representation is called HOJ3D [85], where the 3D space is partitioned into n bins and the joints are associated to each bin using a Gaussian weight function. Thus, a posture is represented by an n-bin histogram and Linear Discriminant Analysis (LDA) is performed to extract the dominant features, having a better discrimination between different classes. Then, a discrete Hidden Markov Model (HMM) is employed to model the temporal evolution of the postures, attained using a clustering algorithm. In addition to k-means clustering, the use of sparse coding has been also proposed for the creation of the codebook. In particular, Luo et al. [86] proposed the DL-GSGC scheme, where the discriminative capacity of the dictionary is improved by adding group sparsity and geometry constraints to the sparse coding representation. A temporal pyramid is adopted to model the temporal information and a linear SVM is chosen as the classification algorithm. Wang et al. [87] firstly considered relations among body joints in the spatial domain, by grouping joints into different

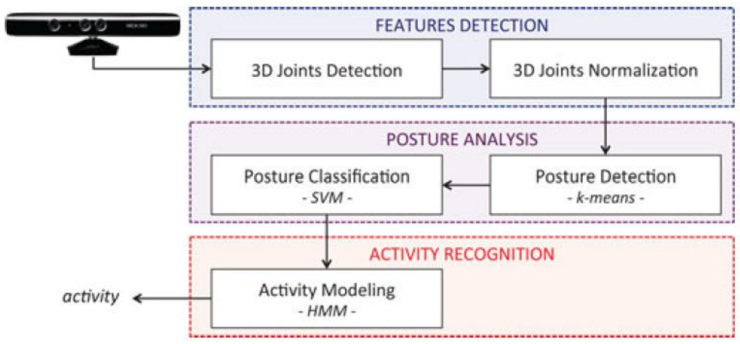


Figure 2.7.: Human activity recognition algorithm constituted by three steps: features detection, posture analysis and activity recognition (reprinted from [90]).

body parts. Then, the temporal relations of the body parts are obtained, and actions are represented by histograms of the detected part-sets. A human action can be characterized also by a combination of static posture features, representing the actual frame, consecutive motion features, computed using the actual and the previous frames, and overall dynamics features, which consider the actual and the initial frames [88]. Principal Component Analysis (PCA) is adopted to obtain EigenJoints from joint differences reducing redundancy and noise. A measurement of Accumulated Motion Energy (AME) has been proposed to quantize the distinctiveness of each frame and to remove the less significant frames. The Naive-Bayes-Nearest-Neighbor (NBNN) classifier can recognize different actions. Taha et al. [89] also exploit joints spherical coordinates to represent the skeleton and a framework composed by a multi-class SVM and a discrete HMM to recognize activities constituted by many actions. Other approaches exploit multiple machine learning algorithms to detect postures and to classify actions. Gaglio et al. [90] divided the process of action recognition in three steps, as shown in Figure 2.7. A set of features is extracted from skeleton joints, then such a set is clustered by applying the k-means algorithm in order to identify the postures involved in each activity. The validation of the postures has been obtained considering a multi-class SVM and a discrete HMM models an activity as a sequence of postures. Also in [91], human actions are considered as a sequence of body poses over time, and skeletal data are processed to obtain invariant pose representations, given by 8 pairs of angles. Then, the recognition is realized using the representation in the dissimilarity space, where different feature trajectories maintain discriminant information and have a fixed-length representation. Ding et al. [92] proposed a Spatio-Temporal Feature Chain (STFC) to represent the human actions by trajectories of joint positions. Before using the STFC model, a graph, called the Actionlets Graph, is used to erase periodic sequences, making the solution more robust to noise and periodic sequence misalignment. Slama et al. [93] exploited the geometric structure of the Grassmann manifold for action analysis. In fact, considering the problem as a sequence matching task, this manifold allows considering an action sequence as a point on its space, and provides tools to make statistical analysis. Considering that the relative geometry between body parts is more meaningful than their absolute locations, a skeletal representation based on 3D relative geometry has been proposed [94]. The relative geometry between two body parts can be described using the rotation and translation required to take one body part to the position and orientation of the other. These rotations and translations required to perform rigid body transformations can be represented as points in a Special Euclidean group $SE(3)$. Each skeleton can be represented as a point in the Lie group $SE(3) \times SE(3) \times \cdots \times SE(3)$, and a human action can be modeled as a curve in this Lie group. A nominal curve for each action category is evaluated using Dynamic Time Warping (DTW) and all the curves are warped to this nominal curve. The warped curves are represented using the Fourier Temporal Pyramid (FTP) representation and a linear SVM is the chosen classifier. The same skeleton feature is also used in [95], where Manifold Functional PCA (mfPCA) is employed to reduce feature dimensionality.

Deep Learning methods, especially Convolutional Neural Networks (ConvNets) and Long Short-Term Memory Networks (LSTMs), have been recently used in different tasks, among wich HAR. Wang et al. [96] proposed a new architecture to use ConvNets, which can automatically learn discriminative features from data, with relatively small datasets. Weighted Hierarchical Depth Motion Maps (WHDMM) allows transforming the problem of action recognition to image classification by converting the spatiotemporal motion patterns into spatial structures. Three WHDMMs are constructed from the projection on each cartesian plane. Three ConvNets are trained on each WHDMM and the results are fused to produce the final classification score. Shahroudy et al. [97] proposed to use the LSTM model which is aware of the body structure. The memory cells of the LSTM are split into part-based subcells and the long-term patterns are learned specifically for each body part. The authors proposed also a new large-scale dataset, which can be efficiently used with data-driven learning methods such as deep learning techniques.

Feature selection methods or optimization strategies may be adopted to improve

Table 2.1.: State-of-the-art datasets for action recognition based on depth or skeletal features, sorted from more quoted to less quoted according to Google Scholar on January 3rd 2017.

Name	**Actions**	**Actors**	**Times**	**Samples**	**Citations**	**Year**
MSR DailyActivity3D [84]	16	10	2	320	614	2012
MSR Action3D [68]	20	10	2 or 3	567	603	2010
UTKinect Action [85]	10	10	2	-	444	2012
MSR ActionPairs [66]	6	10	3	180	338	2013
CAD-60 [98]	12	2+2	-	60	281	2012
CAD-120 [99]	10	2+2	-	120	219	2013
RGBD-HuDaAct [100]	12	30	2 or 4	1189	211	2011
MSRC-12 KinectGesture [101]	12	30	-	594	197	2012
MSR Gesture3D [102]	12	10	2 or 3	336	159	2012
Berkeley MHAD [103]	11	7+5	5	~ 660	110	2013
G3D [104]	20	10	3	-	61	2012
Florence 3D Action [105]	9	10	2 or 3	215	54	2012
ACT4 Dataset [106]	14	24	>1	6844	53	2012
LIRIS Human Activities [107]	10	21	-	-	49	2012
3D Online Action [108]	7	24	-	-	41	2014
UPCV Action [91]	10	20	-	-	39	2014
WorkoutSu-10 Gesture [109]	10	15	10	1500	32	2013
KARD [90]	18	10	3	540	23	2014
UTD-MHAD [110]	27	8	4	861	22	2015
IAS-Lab Action [111]	15	12	3	540	21	2013
KSCGR [112]	5	7	-	-	14	2013
NTU RGB+D [97]	60	40	-	56880	14	2016

the performance of HAR algorithms. These methods may increase the recognition performance because they can select only the relevant features for an efficient discrimination among the activities. Wang et al. [84] proposed a data mining solution to discover discriminative actionlets, which are structures of base features built to be highly representative of one action and highly discriminative compared to other actions. Eweiwi et al. [113] proposed a HAR algorithm exploiting joints where the pose feature is a weighted sum of all joint features. The weights are estimated by

Table 2.2.: Three subsets of actions from MSR Action3D dataset.

AS1	AS2	AS3
[a02] Horizontal arm wave	[a01] High arm wave	[a06] High throw
[a03] Hammer	[a04] Hand catch	[a14] Forward kick
[a05] Forward punch	[a07] Draw x	[a15] Side kick
[a06] High throw	[a08] Draw tick	[a16] Jogging
[a10] Hand clap	[a09] Draw circle	[a17] Tennis swing
[a13] Bend	[a11] Two hand wave	[a18] Tennis serve
[a18] Tennis serve	[a12] Side boxing	[a19] Golf swing
[a20] Pickup & throw	[a14] Forward kick	[a20] Pickup & throw

Partial Least Squares (PLS). Jiang et al. [114] considered the contribution of all the joints and construct informative joint set for each action class, in order to eliminate the redundant joints. The analysis is based on the variation of moving distance of each joint during an action instance and exploits differential entropy. Evolutionary computation has been successfully adopted in feature selection problems, and it has also been considered for the optimization of HAR algorithms. Chaaraoui et al. [115] proposed to use an evolutionary algorithm to find the optimal subset of joints, considering also the topological structure of the skeleton, in order to improve the classification rate. A particular model of evolutionary optimization is represented by the coevolutionary algorithm, which considers several populations: individuals in a population are awarded fitness values based on their interactions with individuals from other populations. Interactions can be competitive, where individuals are rewarded at the expense of those with which they interact, or cooperative, where individual are rewarded if they work well with other individuals [116]. Cooperative coevolutionary algorithms have been also applied to address feature and parameter selection problems in HAR. Chaaraoui et al. [117] proposed an algorithm with three different populations to find the best performing individuals for training instances, skeleton joints and parameters related to the bag of key poses model.

2.2.3. RGB-D Datasets for human action recognition

The growing interest on HAR fostered some researchers to collect HAR datasets and to provide them to the community. Having the same data can help to obtain a fair comparison among different algorithms. Vrigkas et al. [118] identified some issues that should be addressed by a HAR dataset, such as the inclusion of still images and/or video sequences. The data should be high quality and sufficient in terms of amount, including a large number of subjects performing an action and a large number of actions. The inclusion of complex backgrounds and variations in subjects' poses may help to design a robust algorithm. Furthermore, datasets including video data should include changes in illuminations.

A quite large number of datasets for human action (or gesture) recognition have been recorded using RGB-D devices [119, 120], the most used ones are included in Table 2.1. The datasets have been ordered considering the number of citations stated by Google Scholar, that denotes the usage of each dataset by the researchers. In this thesis, some datasets shown in Table 2.1 have been used to evaluate the performance of the proposed HAR algorithms: MSR Action3D, KARD, CAD-60, UTKinect, Florence3D, NTU RGB+D.

The most used one is MSR Action3D [68], released in 2010, constituted by 20 activities performed by 10 actors, 2 or 3 times. In total, 567 sequences of depth (320×240) and skeleton frames are collected using a structured-light depth camera at 15 fps. Considering the skeleton frames, there are 557 sequences effectively available because 10 instances are featured by missing skeletons or they are affected by too many errors. The following actions and gestures are included in the dataset: *high arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pickup and throw.* Due to its complexity, the dataset is usually evaluated considering three different subsets, namely AS1, AS2, and AS3, shown in Table 2.2. Padilla-López et al. [121] reviewed the papers on action recognition considering MSR Action3D dataset and found that there are different evaluation schemes. Even if the most used one is the cross-subject test defined by Li et al. [68], which considers actors 1-3-5-7-9 for training and actors 2-4-6-8-10 for testing, there are also other methods. Some researchers used the cross-subject test with a different splitting: actors 1-2-3-4-5 for training and 6-7-8-9-10 for testing while a more robust evaluation scheme considers all the possible 5-5 splitting (252 combinations).

KARD dataset [90] includes 18 classes that can be divided into 10 gestures (*horizontal arm wave, high arm wave, two hand wave, high throw, draw x, draw tick, forward kick, side kick, bend, hand clap*), and 8 actions (*catch cap, toss paper, take umbrella, walk, phone call, drink, sit down, stand up*). This dataset has been cap-

Table 2.3.: Three activity sets considered from KARD dataset.

Activity Set 1	Activity Set 2	Activity Set 3
[a01] Horizontal arm wave	[a02] High arm wave	[a07] Draw Tick
[a03] Two hand wave	[a10] Side Kick	[a16] Drink
[a12] Bend	[a04] Catch Cap	[a17] Sit Down
[a15] Phone Call	[a07] Draw tick	[a15] Phone Call
[a18] Stand Up	[a13] Hand Clap	[a11] Take Umbrella
[a09] Forward Kick	[a09] Forward Kick	[a08] Toss Paper
[a06] Draw x	[a12] Bend	[a05] High throw
[a14] Walk	[a17] Sit Down	[a01] Horizontal arm wave

tured in a controlled environment, i.e. an office with a static background, and a Kinect device placed at a distance of 2-3 m from the subject. Only some objects, useful to perform some of the actions, were present in the area. The activities have been performed by 10 young people (nine males and one female), aged from 20 to 30 years, with height between 150 and 185 cm. A total number of 540 sequences is provided because each person repeated each activity 3 times. The dataset is composed by RGB and depth frames captured at a rate of 30 fps, with a 640×480 resolution. In addition, 15 joints of the skeleton in world and screen coordinates are provided. Gaglio et al. [90] proposed different evaluation experiments and two modalities of dataset splitting on KARD dataset. In addition to the leave-one-actor-out cross-validation setting, which consists in the adoption of nine actors for training and the tenth for testing averaging the results obtained for each possible combination, the proposed experiments are:

- Experiment A: one third of the data is considered for training and the rest for testing.
- Experiment B: two third of the data is considered for training and the rest for testing.
- Experiment C: half of the data is considered for training and the rest for testing.

The activities constituting the dataset are split in the following groups:

- Gestures (10 classes) and Actions (8 classes).
- Activity Set 1, Activity Set 2, Activity Set 3, as listed in Table 2.3. Similarly to MSR Action3D, Activity Set 1 is the simplest one since it is composed by quite different activities while the other two sets include more similar actions and gestures.

The Cornell Activity Dataset (CAD-60) [98] contains 12 different activities, typical of indoor environments. The activities are *rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking-chopping, cooking-stirring, talking on couch, relaxing on couch, writing on white-board, working on computer*, and are performed in 5 different environments: *bath-room, bedroom, kitchen, living room* and *office*. All the activities are performed by 4 different people: two males and two females, one of whom is left-handed. The actors performed the activities without any specific constraint, the authors simply ensured that the skeleton was correctly detected by Kinect. The dataset is composed by RGB, depth and skeleton data, with 15 joints available. The most used evaluation method is called new-person and, as the leave-one-actor-out cross-validation, consists in the adoption of three actors for training and the fourth for testing.

The UTKinect dataset [85] is composed by 10 activities performed two times by 10 different subjects (9 males and 1 female). The following activities are part of the dataset: *walk, sit down, stand up, pick up, carry, throw, push, pull, wave* and *clap hands*. A number of 199 sequences is available because one sequence is not labeled, and the length of sample actions ranges from 5 to 120 frames. The dataset provides 640×480 RGB frames, and 320×240 depth frames, together with 20 skeleton joints, captured using Kinect for Windows SDK Beta Version, with a final frame rate of about 15 fps. UTKinect dataset has been evaluated using the leave-one-sequence-out cross-validation scheme, where all the sequences except one are used to train the model, and the other is used for testing.

The Florence3D dataset [105] includes 9 different activities: *wave, drink from a bottle, answer phone, clap, tight lace, sit down, stand up, read watch, bow*. These activities were performed by 10 different subjects, for 2 or 3 times, resulting in a total number of 215 sequences. The main difficult of this dataset is that the same action is performed with both hands, and also the presence of very similar actions such as *drink from a bottle* and *answer phone*. The activities were recorded in different environments, and only RGB videos and 15 skeleton joints are available. The evaluation scheme for Florence3D dataset is the leave-one-actor-out cross-validation.

The most recent dataset recorded using RGB-D sensors has been released in June 2016 and is called NTU RGB+D [97]. This is a large-scale dataset featuring a high number of actors and a significant variation of their ages. Furthermore, the dataset has been collected considering many actions (and interactions) and also different views of the same action. NTU RGB+D dataset contains of 56880 RGB+D sequences of 60 different actions performed by 40 different human subjects. The set of 60 actions is constituted by 40 daily actions (such as *drinking water, eating meal/snack, reading, brushing teeth*), 9 health-related actions (such as *sneezing, staggering, falling down*), and 11 interactions (such as *punching, kicking, hugging, handshaking*). The actors are aged between 10 and 35, ensuring a variety in age, gender, and height. Three devices were used to capture at the same time three different horizontal views from the same action ($-45°$, $0°$, $+45°$). To further increase the camera views, on each setup the height and distances of the cameras to the subjects have been changed, with 17 setups and 80 different views. Microsoft Kinect v2 has been used to collect RGB videos (1920×1080), depth maps and infrared frames (512×424), together with skeleton data, represented by 25 joints for each tracked person. Shahroudy et al. [97] defined two evaluation methods:

- cross-subject evaluation: the sequences performed by 20 actors are used as training and the others as testing data. The subjects that have to be used as training are: 1, 2, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 25, 27, 28, 31, 34, 35, 38. Training and testing sets are respectively constituted by 40320 and 16560 sequences but, considering skeleton data, the authors recommend to discard 302 sequences featuring missing or wrong skeletons. Therefore, there are 40091 training sequences and 16487 testing sequences using skeleton joints.
- cross-view evaluation: the sequences captured by different cameras are split between training and testing. The sequences from cameras 2 and 3 are used for training while data from camera 1 provide the testing set. After removing the 302 noisy sequences, the training set has 37646 samples (instead of 37920) and the testing set has 18932 samples (instead of 18960).

Chapter 3.

Depth-based algorithm for side-view mobility assessment

The analysis of the sagittal plane may be relevant for the evaluation of mobility assessment tests, especially because the observation of some angles, in particular the torso angle, can be useful to the fall risk assessment. Microsoft Kinect sensor, using Microsoft SDK [122] or OpenNI libraries [123] with NiTE middleware, allows the extraction of skeleton joints to estimate people's movements. Since these algorithms were designed for gaming purposes, they work quite well when the human is in frontal view but they struggle in the skeleton estimation in side view. In this chapter, an algorithm based on depth data provided by Kinect is proposed to extract 6 joints in the sagittal plane of the human, while is performing the sit-to-stand movement followed by some steps. The performance of the algorithm is compared with the available solutions for markerless joints estimation (Microsoft and OpenNI), considering IR markers as ground-truth. Finally, a further validation process has been carried out with a stereometric system as a reference.

Preliminary results of the method described in this chapter have been published in [124], while a more detailed description of the algorithm and the validation process is included in [125]. The validation of the proposed joint estimation algorithm with a stereometric system has been published in [126].

3.1. Algorithm description

The algorithm exploits only depth data captured by Kinect and extracts x, y, z coordinates of six skeleton joints extracted from the side view of the human: head, shoulder, elbow, hip, knee, ankle. The tracking of the coordinates in subsequent frames allows to describe the movements when the subject stands up from a chair and makes some steps. The system is consituted by three phases, and the global scheme is sketched in Figure 3.1. Depth and RGB frames captured from the three different steps of the global algorithm are shown in Figure 3.2. The three steps are listed here and explained in details in the following sections:

1. construction of a *background depth frame*: as can be noticed from Figure 3.2(a), it represents the test setup, where only the chair is present in the coverage area of the sensor;

2. analysis of the *front-plane pose*: the subject is oriented towards the sensor standing at a fixed distance from it, with outstretched arms and open hands (see Figure 3.2(b)). In this configuration, considering anthropometric models [127], it is possible to detect the joints of interest and to compute the distances between them. The distances information is then used in the next step;

3. *trajectory estimation phase*: the algorithm identifies and tracks the joints while the human subject is performing the test, which starts from sitting position as shown in Figure 3.2(c). The trajectory of the movement performed by each joint is obtained by interpolating the coordinates calculated at each frame.

The choice to split the algorithm in three separated phases may seem a non conventional approach, and partially derives from the choice of not using a machine learning algorithm. In particular, the background acquisition phase helps to extract the human silhouette and, since the test is usually implemented in a controlled environment, it does not limit too much the system applicability. The second phase, where the human shape is analyzed in the front-plane pose, is required to calculate distances between some joints, but it can be avoided once these parameters related to the subject performing the test are known, for example if the same person has been already considered by the system. The global algorithm could be also further simplified removing this phase and introducing by hand the distances among some joints. The third presented phase is required for the estimation of skeleton joints, and it needs the background frame and the joint distances.

3.1.1. Construction of background depth frame

The first phase, which aims to the estimation of the background depth frame, consists in the capture of the test setup. This setup is composed by a Microsoft Kinect sensor placed at an height of 92 cm from the floor and at a distance of 330 cm from a wall. These values are within the range allowed by the manufacturer and represent a tradeoff between pixel density and sensor coverage area. In fact, the

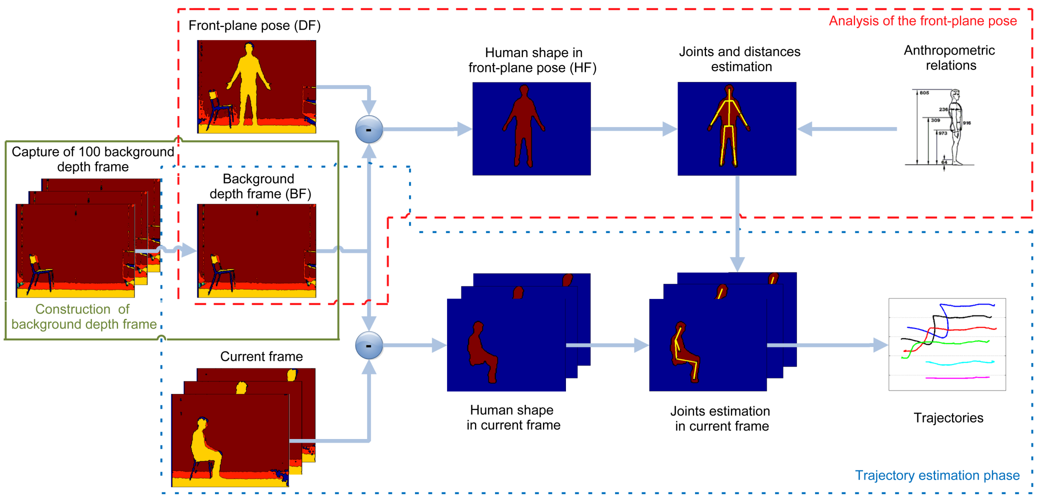

Front-plane pose (DF)
Analysis of the front-plane pose
Human shape in front-plane pose (HF)
Joints and distances estimation
Anthropometric relations
Capture of 100 background depth frame
Background depth frame (BF)
Construction of background depth frame
Current frame
Human shape in current frame
Joints estimation in current frame
Trajectories
Trajectory estimation phase

Figure 3.1.: Main steps of the proposed joints estimation algorithm working in side view.

pixel density would increase locating the sensor closer to the wall, probably leading to an increasing of the accuracy in the estimation of the joints positions. On the other hand, the coverage area would be reduced, restricting the height of people admitted to the test. Kinect is sensitive to reflective surfaces, thus light reflections from the floor, which can lead to wrong depth estimations, can be reduced with a carpet. In order to create a background frame (BF) with a reduced variability of depth data, 100 consecutive depth frames are captured at a rate of 30 fps. The background frame, with a resolution of 320$\times$240 pixels, is obtained by averaging each pixel value in time. The background acquisition process should be performed with only the chair present in the scene, as visible in Figure 3.2(a). The chair should preferably remain in the same position during the following steps of the algorithm. Movements of the chair can generate some artifacts from the background subtraction step, which have to be removed by filtering.

3.1.2. Analysis of the front-plane pose

The estimation of joint distances is performed in front-plane pose, where the subject is placed approximately at a distance of 3 m from the sensor. This configuration is shown in the "*Front-plane pose*" depth frame in Figure 3.1 and also in Figure 3.2(b). Let $DF(x, y)$ be the depth information at the pixel identified by column x and row y in the DF, and $BF(x, y)$ the equivalent in the BF. The foreground frame (FF)

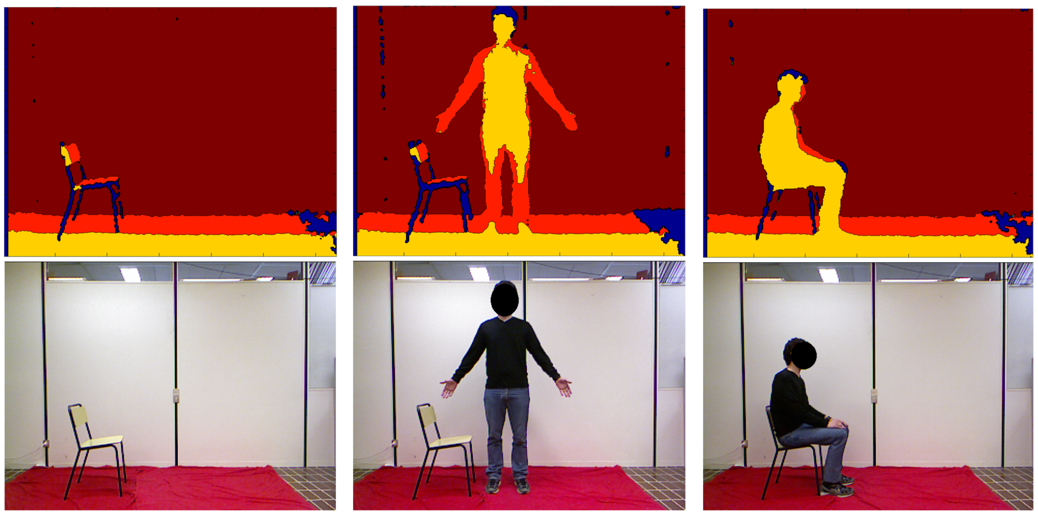

(a) (b) (c)

Figure 3.2.: Depth maps and RGB images of the three main steps of the algorithm. (a) construction of *background depth frame*; (b) analysis of the *front-plane pose*, for the estimation of the distances; (c) *trajectory estimation* tracking (start position).

value in (x, y) is obtained with the application of equation (3.1):

$$FF(x,y) = \begin{cases} 0, & \text{if } |DF(x,y) - BF(x,y)| < Th \\ 1, & otherwise \end{cases} \tag{3.1}$$

Foreground pixels are distinguished from background ones using a threshold (Th) set at a value of 150 mm. Such a threshold has been set to correctly recover the most critical part of the human body that are the feet.

The foreground objects are processed considering the algorithm that finds connected components, and the biggest shape represents the human silhouette, labelled as "*Human shape in front-plane pose*" (HF) in Figure 3.1.

The algorithm processes the human silhouette exploiting anthropometric relations to locate the joints. Then it computes the distances between some of them and, thanks to the rigidity of the skeleton structure, the same distances are used also in the last phase, during the execution of the test. At first, the software needs to identify the head-joint coordinates (J^x_{hd}, J^y_{hd}), and the algorithm implements the steps described in Algorithm 3.1 and in the following on the HF frame:

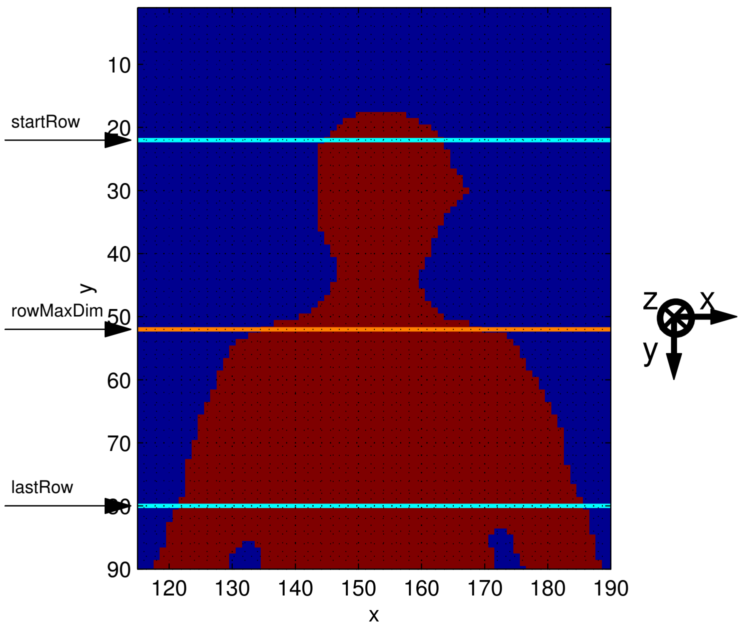


Figure 3.3.: Upper body section of human shape. The algorithm identifies *rowMaxDim*, which represents the top of the shoulders, searching within the interval bounded by *startRow* and *lastRow*.

1. the difference between indices of the bottom and top rows of pixels belonging to the human silhouette represents the parameter *humanHeight*;

2. the head-joint search starts from a specific row of the frame, named *startRow*. This row is located 100 mm below the first row of the silhouette in *HF*. Real world coordinates system $[X_d\ Y_d\ Z_d]^T$ (given as mm) and the depth frame coordinates system $[x_d\ y_d\ 1]^T$ (given as pixels) are related by the equation (3.2):
$$\begin{bmatrix} X_d \\ Y_d \\ Z_d \end{bmatrix} = K_d^{-1} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \lambda \tag{3.2}$$
where K_d is the matrix of intrinsic parameters of the depth camera. Equation (3.2) can be applied knowing the calibration parameters of depth camera and it allows the translation of anthropometric relations, which are calculated in real-world, into distances valid in the depth frame domain, where the coordinate system is represented by pixels;

3. for each analyzed row, the number of columns belonging to the human shape is computed. The row index is increased by 40 mm (*incRow*) every step and the process stops when the index difference between *startRow* and the last row considered (defined as *lastRow*) equals one third of $humanHeight$. Since the idea is to find the shoulders, the stop condition has been set making the assumption that they are located in the upper part of human silhouette. This step is shown in Figure 3.3;

4. the row featuring the maximum difference in the number of computed columns with respect to the previous one is called $rowMaxDim$. This row identifies the top of the shoulders, as can be noticed in Figure 3.3;

5. the quantity $shiftRow$ is defined as the difference between $rowMaxDim$ and the first row that belongs to the human shape ($firstRow$), divided by 3;

6. the y-coordinate of the head-joint (J_{hd}^{y}) is identified by adding the quantity $shiftRow$ to the first row of the human shape;

7. the central pixel of the human shape in J_{hd}^{y} gives the x-coordinate of the head (J_{hd}^{x}).

The row of the frame at which shoulders are located is identified as a shift of 40 mm below $rowMaxDim$. Once the first and the last column occupied by the human silhouette are identified, another horizontal shift of 40 mm, starting from the edges and going towards the inner part of the silhouette, is applied to locate the x-coordinate of left and right shoulders.

The application of anthropometric relations is required to find the hip-joint y-coordinate, knowing the height of the head. The difference between the first row containing the human shape and $rowMaxDim$, reduced by 100 mm which models the neck height, gives the head height. The multiplication of head height and a coefficient c is necessary to obtain the vertical distance between the y-coordinate of the hip and the top of the head. This coefficient depends on the ratio (R) between $humanHeight$ and the head height, and it has been estimated using data in [127], which have been used to construct Table 3.1. Each dataset has an identification number, i.e. 7 for the "AIR FORCE WOMEN-1968" set, 51 for the "ITALIAN MILITARY-1961" set, and so on, which is listed in the first column of Table 3.1. The ratio R, in the fourth column of Table 3.1, is evaluated considering the ratio between Stature (identified by the code 805 in [127]) and the height of the head (code 595). The hip joint distance from the floor can be assimilated as Buttock Height

Algorithm 3.1 Extraction of head joint coordinates in analysis of the front-plane pose.

```
Input: $HF$, $firstRow$, $startRow$, $lastRow$, $incRow$
 1: $diffColMax \leftarrow 0$
 2: $actNumCol \leftarrow 0$
 3: $actRow \leftarrow startRow$
 4: while $actRow \leq lastRow$ do
 5:     $prevNumCol \leftarrow actNumCol$
 6:     $actNumCol \leftarrow$ number of columns of $actRow$ in $HF$
 7:     if $actNumCol - prevNumCol > diffColMax$ then
 8:         $diffColMax \leftarrow actNumCol - prevNumCol$
 9:         $rowMaxDim \leftarrow actRow$
10:     end if
11:     $actRow \leftarrow actRow + incRow$
12: end while
13: $shiftRow \leftarrow (rowMaxDim - firstRow)/3$
14: $J^y_{hd} \leftarrow firstRow + shiftRow$
15: $J^x_{hd} \leftarrow$ central coordinate of $HF$ in $J^y_{hd}$
Output: $(J^x_{hd}, J^y_{hd})$
```

(code 188), which is stated in fifth column. The subtraction of the Buttock Height to the Stature value, whose results are contained in sixth column, gives the distance between the top of the head and the hip joint. The ratio between the quantity in the sixth column and the height of the head results in the coefficient c, which can be used later to calculate the row of the hip knowing the height of the head. The following conditions are obtained by relating the quantity R and the value of c from the data in Table 3.1:

$$c = \begin{cases} 3.2 & R < 6.4 \\ 3.4 & 6.4 \leq R < 7.2 \\ 3.6 & 7.2 \leq R < 7.5 \\ 3.8 & 7.5 \leq R < 8 \\ 4 & 8 \leq R < 8.8 \\ 4.2 & R \geq 8.8 \end{cases} \tag{3.3}$$

Joints from the lower part of the body should also be estimated, and this process starts from the identification of the row where the ankle joint is located. Such a row is found by decreasing the index of the last row of the human shape by 40 mm.

Another anthropometric ratio is used in the computation of the knee. The height

Table 3.1.: Estimation values for the c coefficient that considers anthropometric models.

Dataset	Stature (S) (cm)	Head Height (cm)	R	B. Height (BH) (cm)	$S - BH$ (cm)	c
10	161.3	22.56	**7.15**	84.14	77.16	**3.42**
8	162.77	22.08	**7.37**	82.08	80.61	**3.65**
7	162.1	21.91	**7.4**	82.21	79.89	**3.65**
9	161.92	21.76	**7.44**	82.09	79.83	**3.67**
51	170.62	22.31	**7.64**	86.56	84.06	**3.77**
21	174.72	22.45	**7.78**	89.95	84.77	**3.78**
20	177.1	22.4	**7.9**	91.16	85.94	**3.84**

of the human is multiplied by a coefficient of 0.2 to obtain the distance between the ankle and the knee. This coefficient is also obtained from [127] using the following measurements: Knee Height Sitting (529), Medial Malleolus Height (579) and Popliteal Height (678).

The area of the left arm of the person is processed to extract the remaining joints. Once the left arm is found, by looking for a gap in the column indices in the left part of the body, the left wrist is located considering a distance of the half of head height to the ends of the hand. The distance between shoulder and wrist is computed, and the elbow can be located in the middle of this range.

The next step of the algorithm exploits the following distances: head-shoulder (HS_dist), shoulder-elbow (SE_dist), elbow-wrist (EW_dist), ankle-knee (AK_dist) and knee-hip (KH_dist).

3.1.3. Trajectory estimation phase

The third step starts with the identification of the human shape in the depth frame, as described in the previous phase related to the analysis of the front-plane pose.

The distance quantity named $shiftRow$ is exploited to find the the head joint, starting from the first row belonging to the human. The joint is placed in the middle of the shape in the row derived by averaging the pixel column indices that belong to the human silhouette.

In order to locate the remaining joints, it is necessary to extract only the side of

Algorithm 3.2 Extraction of shoulder joint coordinates in trajectory estimation phase.

```
Input: $HF\_side$, $HS\_dist$, $(J^x_{hd}, J^y_{hd})$
1: $J^y_{sh} \leftarrow J^y_{hd} + HS\_dist$
2: $J^x_{sh} \leftarrow$ central coordinate of $HF\_side$ in $J^y_{sh}$
3: $\alpha \leftarrow \tan^{-1} \frac{|J^x_{hd} - J^x_{sh}|}{|J^y_{hd} - J^y_{sh}|}$
4: $HS\_shift_x \leftarrow HS\_dist \cdot \sin(\alpha)$
5: $HS\_shift_y \leftarrow HS\_dist \cdot \cos(\alpha)$
6: $J^x_{sh} \leftarrow J^x_{hd} - HS\_shift_x$
7: $J^y_{sh} \leftarrow J^y_{hd} + HS\_shift_y$
Output: $(J^x_{sh}, J^y_{sh})$
```

the human silhouette which is closest to the sensor. This allows to reduce the area of the human, considering that the joints should be placed only on one side of the person, thus avoiding the detection of the leg which is closer to the wall while the person is walking. The side selection requires initially to find a depth value of the human closest side, which can be obtained through multiple steps. One row of the leg, exactly in between the ankle and knee joints, has been considered to extract the lowest depth value. With the aim of mitigating fluctuations due to noise, a square sub-matrix of the AK_dist side is selected around the pixel showing the minimum value. In this region, the pixels for which the difference of their depth value from the depth value of the central pixel overcomes the *sideManThreshold* (80 mm), are filtered. By taking into account the remaining pixels, the average depth is calculated and this value is set as the distance between the human and the sensor. Based on this distance, the frame containing the side part of the human shape (HF_side) is estimated removing the pixels for which the depth value overcomes the *sideManThreshold* value.

The distance HS_dist is used to find the shoulder joint, identified by (J^x_{sh}, J^y_{sh}), from the head, initially assuming a vertical alignment with the two joints. After the identification of the vertical coordinate, the corresponding column is computed by averaging the first and last columns belonging to the human shape. The estimation is then improved to take into account a possible inclination of the torso, which can be relevant especially during the sit-to-stand phase. In fact, the segment connecting the joints needs to be equal to HS_dist considering the head-shoulder angle. This process is described in Algorithm 3.2.

Only the pixels belonging to the arm are selected to extract the elbow joint. This

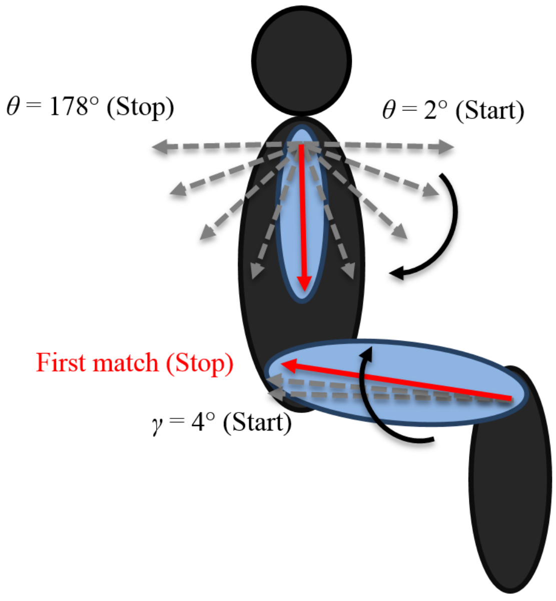


Figure 3.4.: Elbow and hip identification procedure. Two rotating vectors, anchored in the shoulder and knee joints, are exploited to identify elbow and hip coordinates respectively.

selection process follows the same approach used to extract the side of the human shape. Considering only the arm area and starting from the shoulder joint, a vector of length SE_dist is rotated from $\theta = 2°$ to $\theta = 178°$, as shown in Figure 3.4. The number of overlapping pixels between the vector and the arm region is calculated at each step. The orientation featuring the maximum number of matching pixels is chosen as the final orientation of the vector, and the elbow position is therefore extracted.

The joints belonging to the lower part of the body are estimated independently from the upper part. The ankle joint is located similarly to the head joint, by applying a shift from the last row belonging to the human. The knee joint is positioned with a first vertical identification of the *kneeRow* by AK_dist and a subsequent adjustment, by considering the angle between knee and ankle, in a similar manner to what has been done for the shoulder joint (see Algorithm 3.2). The hip joint estimation, similarly to the process required by the elbow, exploits a vector of length KH_dist, anchored in the knee joint, which rotates from $\gamma = 4°$ to the best matching angle by $2°$ steps, as shown in Figure 3.4.

At the end of the described process, the coordinates of the six skeleton joints are

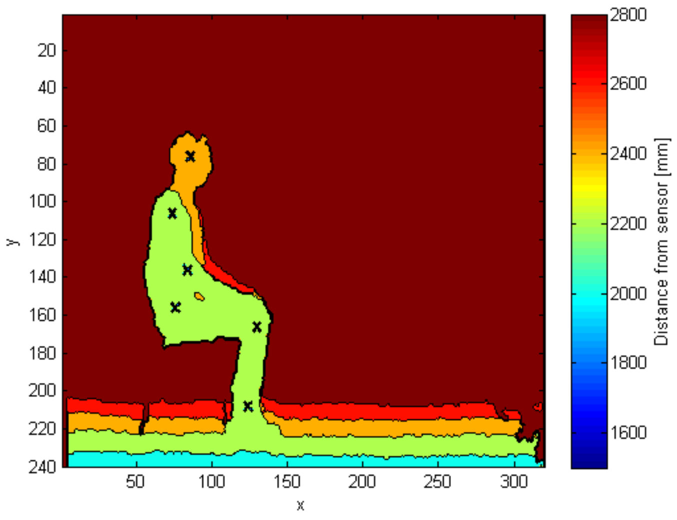


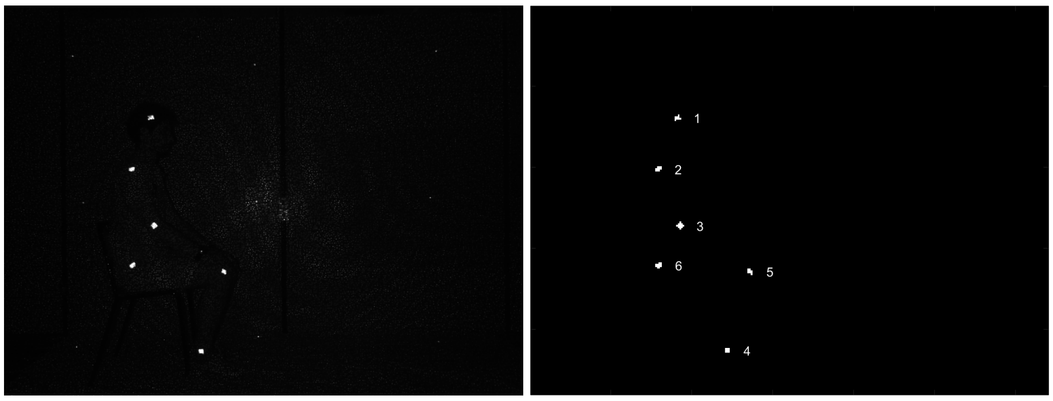
Figure 3.5.: Depth frame in side-view where the six estimated joints are positioned.

known, and they can be represented in the depth frame. Figure 3.5 shows the side view of the human shape in the depth map, before the execution of the test. Each joint is represented by a cross: starting from the top it is possible to identify head, shoulder, elbow, hip, knee and ankle joints.

The execution time of the MATLAB implementation on a depth frame with a resolution of 320×240 can give an idea of the computational requirements of the algorithm. A desktop PC equipped with an Intel i7 Windows 8.1 and 8GB RAM can process a depth frame in 51 ms, in average. Considering that the Kinect sensor provide frames with a rate of 30 Hz, an optimized C++ implementation could run at real time.

3.2. Performance analysis

The performance of the algorithm has been validated considering a marker-based algorithm as a reference, and by comparing the accuracy in the identification of the six joints with other markerless estimation algorithms, provided by Microsoft SDK (Algorithm 1) and OpenNI libraries with NiTE middleware (Algorithm 2). Since the Microsoft Kinect sensor allows capturing infrared data, IR reflective sticky markers have been attached on the human body to identify the real joint positions. The infrared frame with IR markers placed on the body of a subject ready to perform



(a) (b)

Figure 3.6.: (a) Infrared frame captured by the Kinect sensor showing a human subject with IR active sticky markers identifying the joints. (b) Blobs extracted from the IR frame, associated to joints coordinates.

the test is shown in Figure 3.6(a). The positions of the six considered joints are easily identifiable by applying a blob detection algorithm, as shown in Figure 3.6(b), where each blob is associated to a joint index.

The procedure to validate the algorithm has been performed in a laboratory environment, as shown in Figure 3.2. However, the same setup may be reproduced in any other indoor environment where the aforementioned distance conditions are respected. The validation experiment requires the execution of the two preliminary steps of the joints estimation algorithm, which consist in the creation of the background depth frame and also the computation of distances among the parts of the subject's body. Then, to validate the test execution, the IR active sticky markers that identify the joints are applied on the subject. When the subject is seated on the chair and ready to perform the test, the data acquisition process starts. During the test execution, the subject stands up and steps forward along a distance of a few meters. The data acquisition stops when the subject has performed some steps and she is standing. A sequence of frames extracted from one execution of the test is shown in Figure 3.7.

A specific acquisition tool, which is called *Skeletal viewer* and detailed in Chapter A, has been developed to capture multiple streams from Kinect simultaneously. This software exploits Kinect for Windows SDK to capture and store the following data streams:

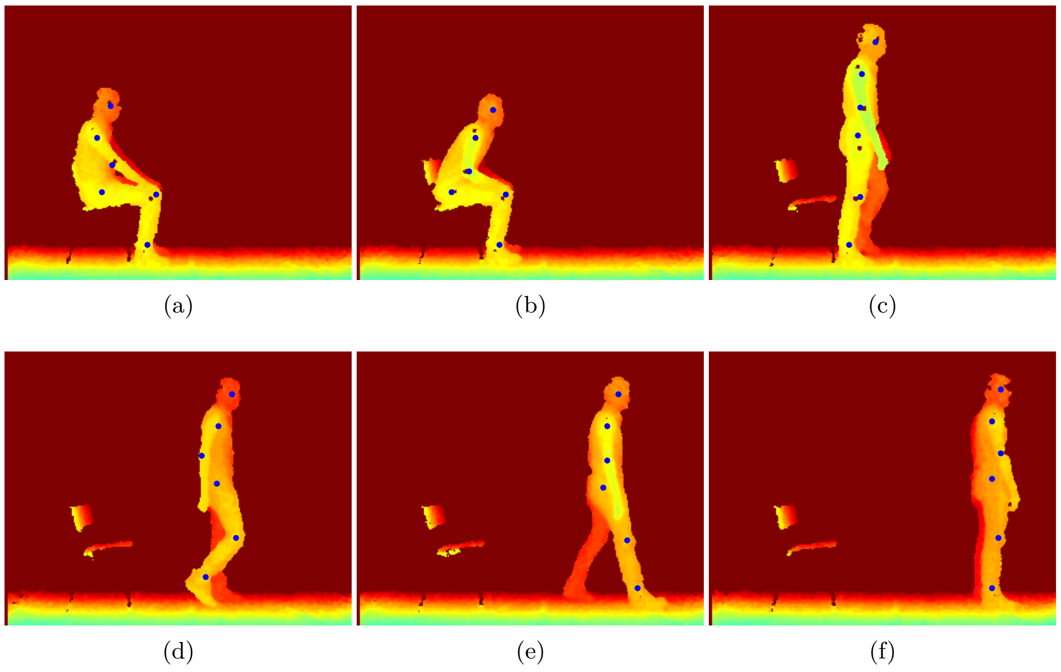


Figure 3.7.: Sequence of frames extracted from one execution of the test. The blue circles are the joints estimated by the Proposed algorithm.

- depth frames, used by the Proposed algorithm and Algorithm 2 to compute joint coordinates;
- IR frames, used to extract the joint positions identified by markers, which represent the ground-truth;
- skeleton frames, computed by Algorithm 1 and used for comparison.

The trajectories of the joints estimated by the four tracking systems can be compared by evaluating the euclidean distance among the joint coordinates in the frame domain of dimensions 320 × 240 pixels. Being $J^x_{i,m}(k)$ and $J^y_{i,m}(k)$ the x and y coordinates of the i-th joint in the k-th frame for the marker-based method (m) and being $J^x_{i,p}(k)$ and $J^y_{i,p}(k)$ the same coordinates obtained by the Proposed algorithm (p), the magnitude of the difference vector $D_i(k)$ is given by:

$$D_i(k) = \sqrt{\left(J^x_{i,m}(k) - J^x_{i,p}(k)\right)^2 + \left(J^y_{i,m}(k) - J^y_{i,p}(k)\right)^2} \qquad (3.4)$$

The computation can be repeated for the entire test, which is constituted by multiple

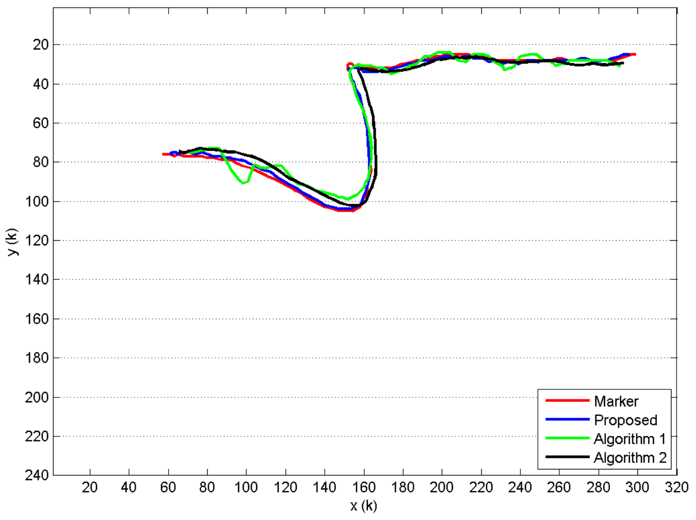


Figure 3.8.: Head joint trajectories revealed by the analyzed algorithms. The trajectories are constituted by the (x, y) joint coordinates at each frame.

depth frames (K), and it is possible to define a vector $\mathbf{D}_i$ for the i-th joint:

$$\mathbf{D}_i = [D_i(1), D_i(2), \cdots D_i(K)], \qquad i = 1, 2, \ldots 6 \tag{3.5}$$

The same quantities can be computed for Algorithm 1 and Algorithm 2 and used for comparison.

The Proposed approach shows a better trajectory estimation in most of the cases, if compared to Algorithm 1 and Algorithm 2. Figure 3.8 shows the trajectories of the head joint during a single execution of the test. The curve named Marker, revealed by the IR marker trajectory, is better approximated by the Proposed one, in comparison to Algorithm 1 and Algorithm 2.

The differences $\mathbf{D}_i$ provided by the three analyzed markerless systems with respect to the Marker system, for all of the joints estimated during the same test execution, are shown in Figure 3.9. The head joint estimation given by the Proposed solution is featured by a lower error, as can be noticed also in Figure 3.9(a), where smaller values and reduced variability of the vector $\mathbf{D}_1$ affect the Marker-Proposed curve. The algorithm Proposed does not suffer from the fluctuations that affect Algorithm 1 and 2. In particular, considering the shoulder joints shown in Figure 3.9(b), large errors for the Marker-Algorithm 2 curve are revealed when the subject is getting up from the chair, and he is tilted forward (around frame index 35). The Marker-Algorithm 1 curve is also featured by large errors during the sit-to-stand in the lower

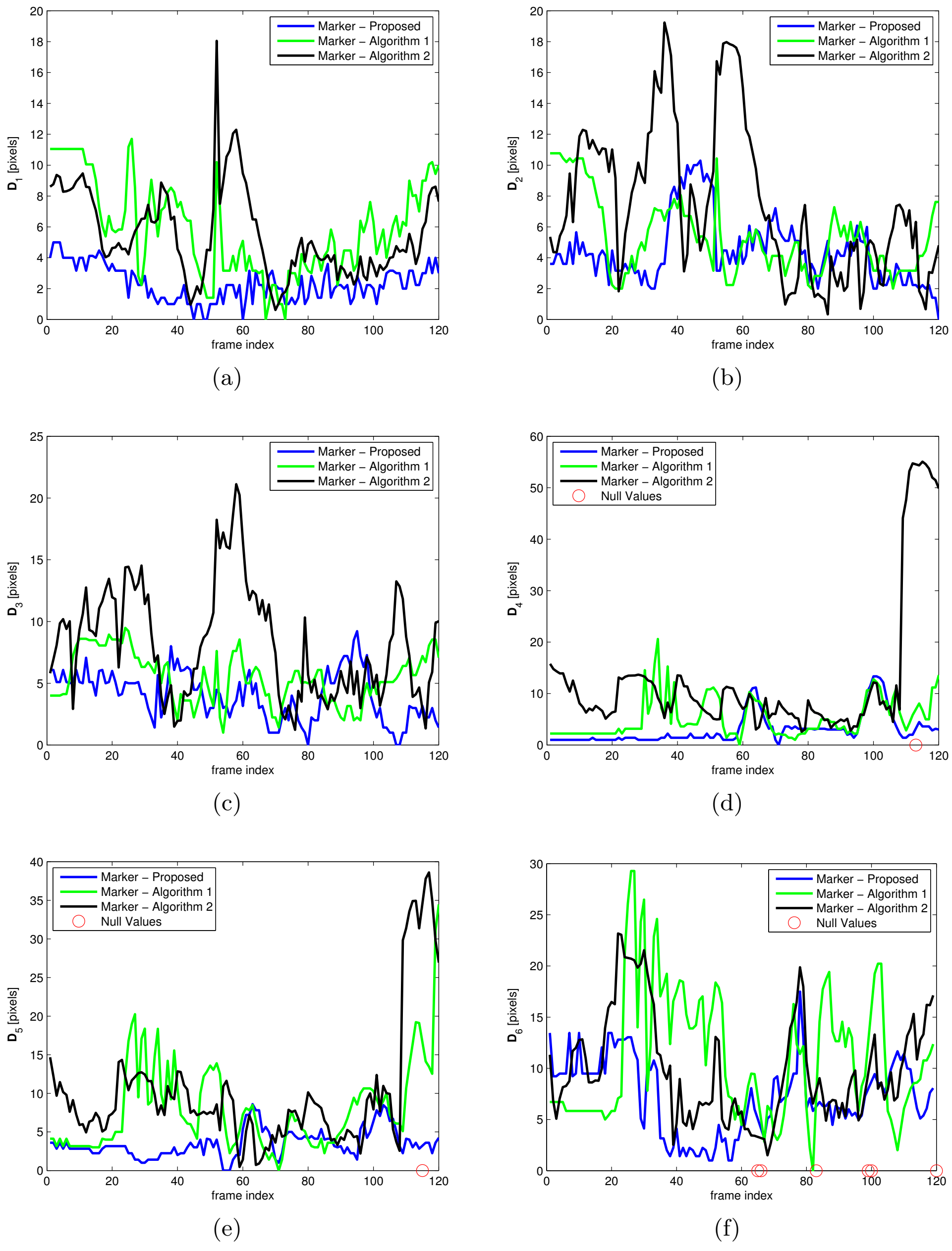


Figure 3.9.: Magnitude values of the difference vectors $\mathbf{D}_i$ computed for different joints: (a) head, (b) shoulder, (c) elbow, (d) ankle, (e) knee and (f) hip, evaluated over a test execution.

Table 3.2.: Statistical parameters associated with differences $\mathbf{D}_i$, expressed in pixels.

Joints	Marker-Proposed		Marker-Algorithm 1		Marker-Algorithm 2	
	μ	σ	μ	σ	μ	σ
head	**2.33**	**1.14**	5.89	3.02	5.52	2.83
shoulder	**4.42**	**2.14**	5.35	2.44	7.44	4.89
elbow	**4.02**	**2.00**	5.46	2.05	8.17	4.50
ankle	**3.04**	**3.00**	5.27	3.86	12.30	13.11
knee	**3.51**	**1.80**	7.94	5.71	9.95	8.10
hip	**7.25**	**3.70**	11.07	6.32	10.05	5.21

part of the body, shown in Figures 3.9(d–f). Ankle and knee joints are also subjects to wrong estimation in the final part of the test, as shown in the Marker-Algorithm 2 curve of Figures 3.9(d,e).

One or more IR markers used for validation can get hidden to the sensor due to some movements of the subject. If this condition happens, as in Figures 3.9(d–f), the position error cannot be evaluated, and a null value is considered in the corresponding frame of $\mathbf{D}_i$ vector.

Considering each vector $\mathbf{D}_i$, its average (μ) and standard deviation (σ) may be assumed as global performance parameters of the algorithm. Table 3.2 shows the statistics of the different algorithms evaluated against the marker-based one, for the same test realization. The Proposed system allows to have lower values of μ and σ for all the estimated joints. It shows the best performance for the standard deviation of the head joint, while the hip joint has the highest average value and variability in the set of joints analyzed.

The different algorithms are finally tested on a dataset of 18 test executions performed by different subjects, with a range of human heights from 1.6 to 1.85 meters and various body sizes. Figure 3.10 shows the mean μ and standard deviation σ for each joint, averaged over 18 test executions. The considerations for one test made looking at Table 3.2 can be also derived looking at Figure 3.10 summarizing the results. Head and shoulder joints are characterized by low error values while ankle, knee and hip joints show larger errors. The Proposed algorithm is featured by lower average and standard deviation values for all the joints, with a bigger error featuring the hip joint. The most important noise is in the estimation of the ankle joint for

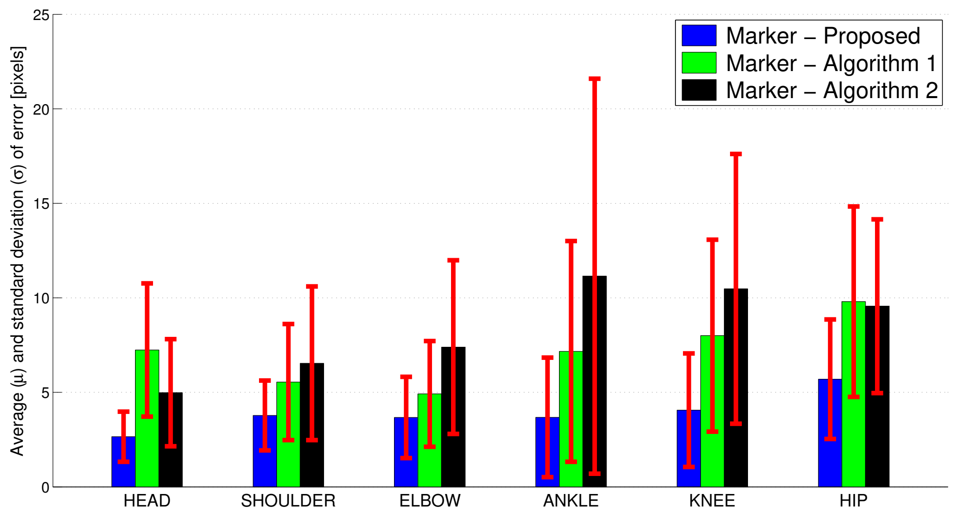


Figure 3.10.: Performance comparison of the Proposed algorithm, Algorithm 1 and Algorithm 2 versus the marker-based system over 18 test realizations.

Algorithm 2.

3.3. Validation with stereometric system

A validation procedure has been carried out considering the accuracy in the estimate of knee and hip joint angular values during the sit-to-stand (STS) test. The reference system is represented by the the gold-standard approach for 3D kinematic analysis: an optoelectronic stereometric system constituted by six cameras (SMART-D, BTS). The 3D positions of passive retroreflective markers placed on suitable anatomical landmarks of the body surface can be obtained with this system.

The system setup for this validation process is similar to the one adopted in the analysis of the performance with the IR sticky markers. The presence of a stereometric system providing ground-truth coordinates of markers is the main difference. The validation process is performed only for the sit-to-stand phase (STS) and the evaluation metric is the error in the estimation of torso and knee angles calculated from the joint coordinates. For each realization of the STS test, the following data are recorded:

- coordinates of markers, provided by the stereometric system, consisting in a set of x, y, z coordinates revealing the position of each IR marker;

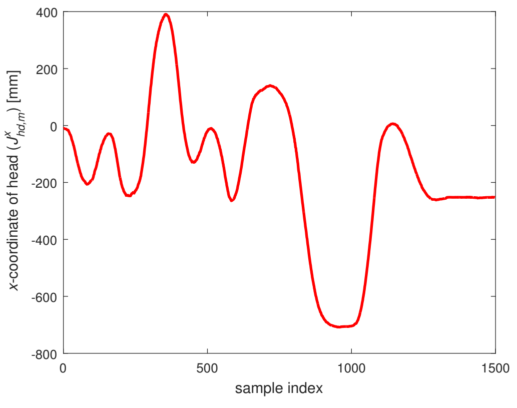
- depth frames, provided by Microsoft Kinect sensor, which are used by the Proposed joint estimation algorithm and by the OpenNI one (Algorithm 2);
- coordinates of the skeleton joints, estimated by the Microsoft SDK algorithm (Algorithm 1) exploiting Kinect sensor.

The *Skeletal viewer* tool has been used to capture data from Kinect v1 and the proprietary software provided by the manufacturer of the marker-based system has been also used during the data acquisition process.

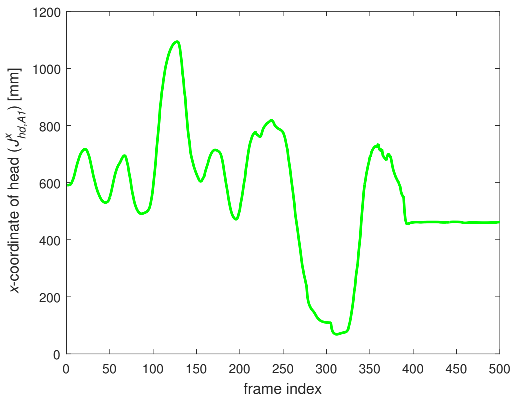
3.3.1. Data synchronization

The computation of a performance index needs the realization of a correspondence between samples of different systems, which requires to solve some issues. Firstly, it is not possible to synchronize the data when they are received by the machine because the two systems involved (Kinect and stereometric system) are running on different platforms. This implies that the starting capture time of marker-based and Kinect-based algorithms are slightly different. The recorded data do not start when the subject is seated on the chair and ready to perform the test, but when she is standing in front of the Kinect sensor. This procedure is required to use the OpenNI library to compare the performance because it works offline on captured depth data and it needs to see the human in frontal view to start the skeleton tracking. Therefore, there is the need to discard the initial portion of the captured data before the evaluation process. Secondly, the acquisition systems have different sampling rates: the stereometric system captures marker coordinates with a rate of 100 Hz, while the Kinect-based application provides depth frames at approximately 30 fps.

Solving the former issue requires finding the time indices in the sets of marker coordinates, depth frames and SDK joints coordinates, that represent the starting points of the STS test. The different data streams captured from Kinect are synchronized to each other by using frame timestamps, so it is sufficient to associate the frame index of the marker-based system to the corresponding index of a frame obtained from Kinect. The idea consists in the correlation of the head joint trajectories in both systems (Kinect and Marker). The choice of the joint/marker of the head derives from the consideration that it represents the best identified skeleton point, and also from the fact that it features a movement that can be easily detected. Due to the setup configuration and adherence of marker and SDK skeleton coordinates systems, the frames of interest for the STS test occur after the minimum of the



(a)



(b)

Figure 3.11.: (a) Trajectory of head marker x-coordinate ($J^x_{hd,m}$). (b) Trajectory of head joint x-coordinate extracted by Algorithm 1 ($J^x_{hd,A1}$).

x-coordinate of head marker/joint. Figure 3.11 shows the values of the x-coordinate of head in the marker-based system ($J^x_{hd,m}$) and in the Microsoft SDK one ($J^x_{hd,A1}$), for all the recorded frames of one test execution. Figure 3.11(a) shows that the minimum of $J^x_{hd,m}$ is near to the sample 1000, while Figure 3.11(b) denotes a minimum of $J^x_{hd,A1}$ around the frame 300.

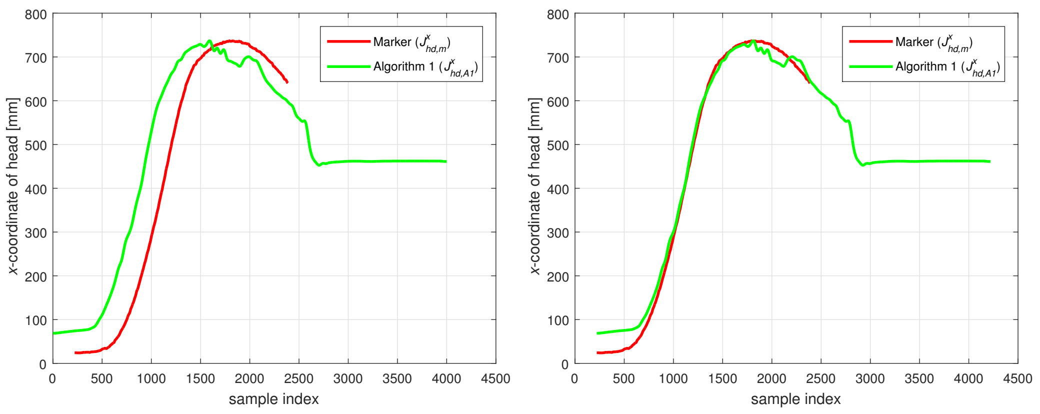
The offset computation between the two trajectories requires additional steps:

- extraction of the remaining portions of the curves, starting from the found indices, and in their interpolation, to reach the same sampling rate of 1 kHz.
- overlap of the interpolated curves and computation of the error, defined as the euclidean distance, for different shift values;
- definition of the offset between the markerless and the marker-based systems as the shift value that results in the minimum error.

Figure 3.12 shows the overlap of $J^x_{hd,m}$ and $J^x_{hd,A1}$, both interpolated and starting from their respective minimums. As the Figure 3.12(a) shows, there is an offset between the two trajectories, that can be compensated after its evaluation, as in Figure 3.12(b).

More in detail, the procedure for the evaluation of the time offset, is composed by multiple steps:

1. Selection of the portion of interest of the marker-based curve. Considering the



(a) (b)

Figure 3.12.: Overlap of head marker/joint x-coordinate after interpolation and amplitude scaling. The overlap is realized before (a) and after (b) offset compensation.

$J^x_{hd,m}$ vector, with a length of N samples, the sample n_{max_m} is defined as:

$$J^x_{hd,m}(n_{max\_m}) = \max\left\{J^x_{hd,m}(n)\right\},\ n = 1, 2, \ldots, N \tag{3.6}$$

The first sample (n_1) that satisfies (3.7) is the starting point of the selected curve, and it is given by:

$$n_1 = \min\{n\} \quad | \quad J^x_{hd,m}(n) - J^x_{hd,m}(1) > Th_1, \quad n = 2, 3, \ldots, n_{max\_m} \tag{3.7}$$

where the threshold $Th_1 = \left(J^x_{hd,m}(n_{max\_m}) - J^x_{hd,m}(1)\right)/10$ represents the 10% of the x-coordinate range $[J^x_{hd,m}(1), J^x_{hd,m}(n_{max\_m})]$. The end sample of the selected curve is represented by the last sample (n_2) that satisfies the following relationship:

$$n_2 = \max\{n\} \quad | \quad J^x_{hd,m}(n) - J^x_{hd,m}(N) > Th_2, \quad n = n_{max\_m}, \ldots, N \tag{3.8}$$

where $Th_2 = \left(J^x_{hd,m}(n_{max\_m}) - J^x_{hd,m}(N)\right)/10$.

2. Computation of the euclidean distance for different shifts. By defining as n_{max_A1} the sample showing the maximum value of $J^x_{hd,A1}$ (equivalent of n_{max_m}

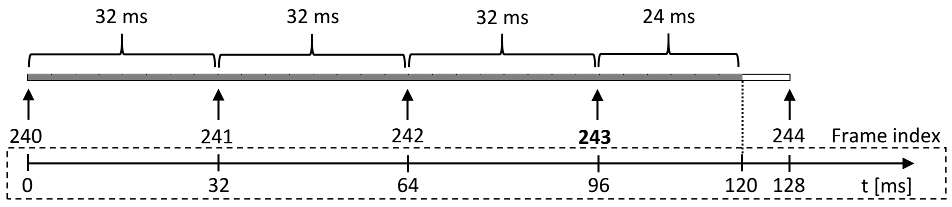


Figure 3.13.: Time axis in offset compensation.

defined in equation (3.6)), the quantity M can be defined as:

$$M = n_{max\_m} - n_{max\_A1} \tag{3.9}$$

and it is used to set the amplitude of shifts, which goes from 0 to $2M$. This interval certainly allows to reach the overlap between the maximum of the two curves. For each offset value (n_{sh}), the euclidean distance between the trajectories is evaluated as follows:

$$d_{sh} = \sqrt{\sum_{n=n_1}^{n_2} |J^x_{hd,A1}(n) - J^x_{hd,m}(n - n_{sh})|^2}, \quad n_{sh} = 0, 1, \ldots, 2M \tag{3.10}$$

3. Evaluation of the shift that gives the minimum euclidean distance:

$$n^*_{sh} = n_{sh} \quad | \quad d^*_{sh} = \min\{d_{sh}\} \tag{3.11}$$

The offset between the two systems has been evaluated on the interpolated data, which have a resolution of 1 ms, while the Kinect-originated data have a resolution variable between 32 and 37 ms. Assuming that the minimum value of $J^x_{hd,A1}$ corresponds to the frame index 240 and the found offset is 120 ms, the entire offset cannot be covered simply by changing the index of the frame from which the processing starts. A sample situation is shown in Figure 3.13 where the considered starting index is 243, which allows to compensate 96 ms. The remaining delay of 24 ms will be covered in the comparison phase, when the trajectories of angles between joints will be interpolated, using the same approach.

3.3.2. Results and discussion

The passive IR markers used by the stereometric system have been positioned as shown in Figure 3.14(a), plus an additional marker placed on the top of the head,

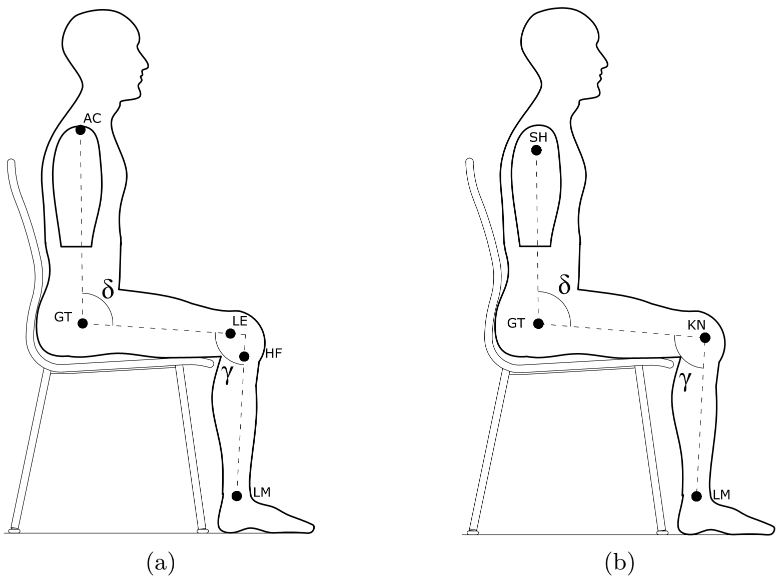


Figure 3.14.: (a) Anatomical landmarks used by the stereometric marker-based system. (b) Skeleton joints by markerless algorithms used for angles computation.

used for synchronization. The markers correspond to Shoulder Acromion (AC), Great Trochanter (GT), Lateral Epicondyle (LE), Head of Fibula (HF), Lateral Malleolus (LM). Angles in the sagittal plane δ and γ are considered for comparison purpose. The process of angles computation is slightly different because the reference points in the marker-based system do not correspond exactly to the joints estimated by Kinect algorithms, shown in Figure 3.14(b). The joints involved in the computation of δ and γ angles are labeled as Shoulder (SH), Great Trochanter (GT), Knee (KN), Lateral Malleolus (LM).

The angle δ evaluated by the marker-based system at the k-th frame can be defined as $\delta_{m,k}$. Considering a number of K frames that constitutes the entire test sequence, the vector $\boldsymbol{\delta}_m$, that considers all the angles of the test is given by:

$$\boldsymbol{\delta}_m = [\delta_{m,1}, \delta_{m,2}, \dots \delta_{m,K}] \tag{3.12}$$

By extending the same definition for the other Kinect-based algorithms, the vectors $\boldsymbol{\delta}_p$, $\boldsymbol{\delta}_{A1}$ and $\boldsymbol{\delta}_{A2}$ can be defined for the Proposed algorithm, the Microsoft SDK-based solution (Algorithm 1) and for the OpenNI one (Algorithm 2) respectively. The error vector $\boldsymbol{\Delta}_{m,p}$ is defined as the absolute difference of the angles computed

Table 3.3.: Statistical indices of angle errors (in degrees). Errors greater than 100° are not included.

Test-ID	Angle	$\mathbf{\Delta}_{m,p}$		$\mathbf{\Delta}_{m,A1}$		$\mathbf{\Delta}_{m,A2}$	
		μ	σ	μ	σ	μ	σ
$P1_1$	δ	**5.9**	**3.6**	8.0	5.5	10.3	6.8
	γ	**4.1**	**1.8**	16.5	20.1	13.4	8.5
$P1_2$	δ	**8.9**	**5.9**	27.4	24.9	15.1	7.2
	γ	**12.3**	**5.8**	17.3	23.6	15.0	26.0
$P1_3$	δ	**5.4**	**3.1**	23.7	24.6	25.1	27.8
	γ	**6.2**	**6.3**	20.4	35.0	22.0	28.8
$P1_4$	δ	**4.8**	**2.1**	27.6	34.9	13.1	10.2
	γ	**5.7**	**7.4**	25.9	28.2	12.0	6.2
$P2_1$	δ	**4.0**	**3.2**	36.0	43.0	–	–
	γ	**3.6**	**2.5**	53.5	36.7	–	–
$P2_2$	δ	**5.9**	**3.3**	17.5	7.6	–	–
	γ	**1.9**	**1.5**	20.0	25.9	–	–
$P2_3$	δ	6.2	**2.7**	20.1	15.1	**5.4**	3.2
	γ	**1.6**	**1.1**	19.5	19.2	6.7	4.4
$P2_4$	δ	**5.4**	**3.5**	15.3	8.1	60.8	53.3
	γ	**2.6**	**1.7**	11.5	7.6	15.5	12.5

by the reference system and the Proposed algorithm:

$$\mathbf{\Delta}_{m,p} = |\boldsymbol{\delta}_m - \boldsymbol{\delta}_p| \tag{3.13}$$

The same reasoning brings to the definition of $\mathbf{\Delta}_{m,A1}$ and $\mathbf{\Delta}_{m,A2}$ that can be compared with classic statistical indexes, as average (μ) and standard deviation (σ). The same indices are evaluated for angle γ and the results have been obtained considering 4 runs of the STS test, performed by 2 healthy and young subjects. Results are given in Table 3.3, where the Test-ID Pi_j refers to the j-th repetition performed by the i-th subject.

The following considerations, related to average and standard deviation values, can be done on the results shown in Table 3.3:

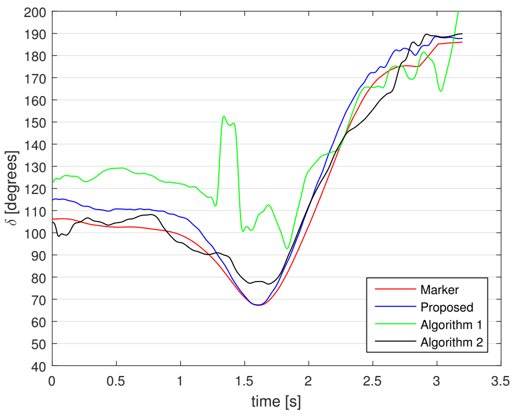
- the Proposed algorithm is featured by lower average error values when compared to Algorithm 1 and Algorithm 2 in almost all the test executions. The only case when Algorithm 2 performs better is for δ angle of $P2_3$. The average error value is below ten degrees both for δ and γ angles in all the cases, except for the γ angle of $P1_2$, which is 12.3°.
- The standard deviation (σ) of the Proposed algorithm is lower than the other markeless solutions in all the executions. The best σ value is reached in the estimation of γ angle in the test $P2_3$ and it is 1.1°. The worst value (7.4°) is also in the evaluation of the γ angle, in test $P1_4$.

The performance analysis of other Kinect-based algorithms from Table 3.3 demonstrates that Algorithm 1 and Algorithm 2 are characterized by large values in terms of average and standard deviation. Figures 3.15(a,c,e) show results relative to the best performance test ($P2_3$) while Figures 3.15(b,d,f) are related to the worst one ($P1_2$). Trajectories of δ and γ angles have been smoothed considering a 150-points moving average filter.

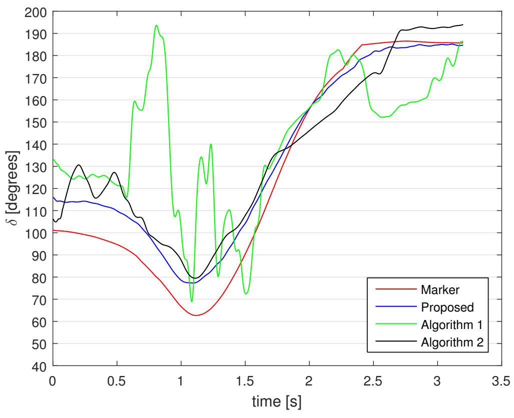
Looking at Figure 3.15(a) and 3.15(c), δ and γ angles computed using Algorithm 1 are affected by large errors, especially in the central part of the test, when the person is tilted forward. The Proposed algorithm is not affected by large fluctuations and is able to approximate the reference curve quite well, especially in γ angle. An error of about 10° is provided by Algorithm 2 when the person is tilted forward. Figure 3.15(e) shows that the δ angle of the Proposed algorithm has an error of about 10° in the first and in the last part of the test, while it is very low when the person is tilted forward. The error related to γ angle is lower than 5° during the entire test.

Considering the worst test, the δ angle (shown in Figure 3.15(b)) is featured by a larger error in the first part of the test while the error in the γ angle increases in the final part, as can be seen from Figure 3.15(d). Looking at Figure 3.15(f), even if the average errors are about 9° for δ and 12° for γ angles, the Proposed algorithm can still perform better than Algorithm 1 and 2.

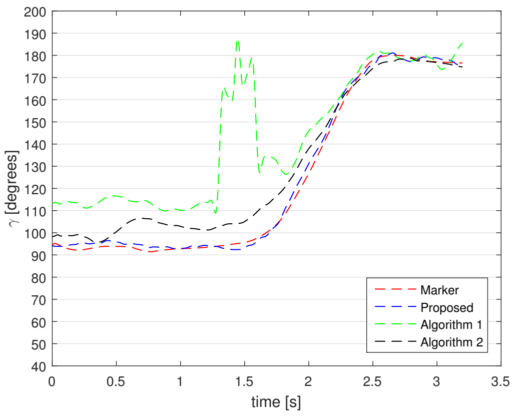
By comparing the results for γ and δ angles, the Proposed algorithm is usually able to better estimate γ. The little displacement of the shoulder joint considered by the marker-based and markerless models shown in Figure 3.14 can be the reason of this fact. In average, for all the tests shown in Table 3.3, the Proposed algorithm is characterized by an error of 5.8° for the δ angle and 4.7° for the γ angle.



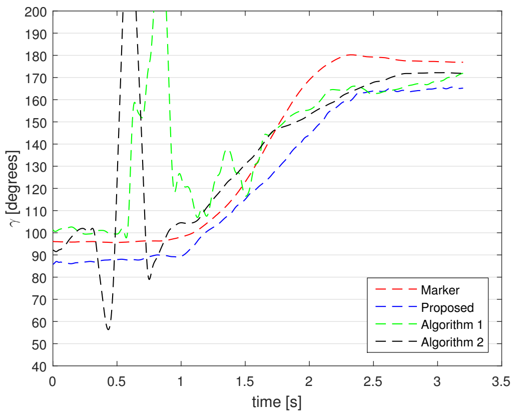
(a)



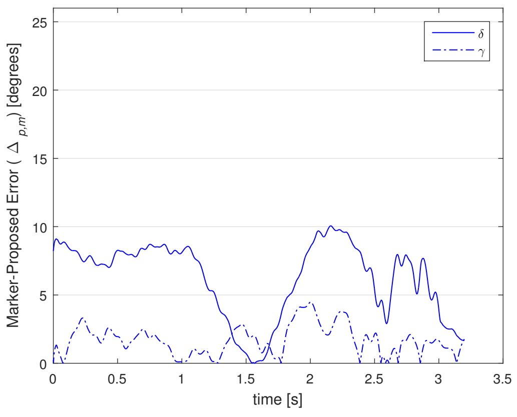
(b)



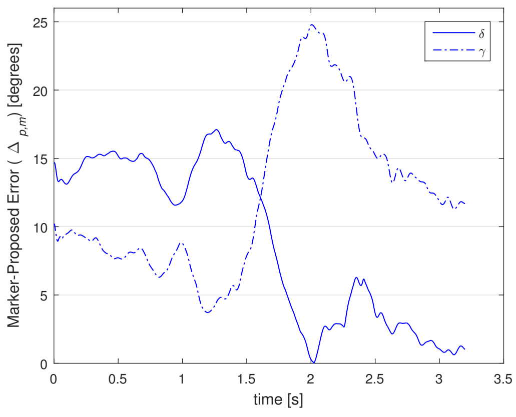
(c)



(d)



(e)



(f)

Figure 3.15.: Trajectories of δ and γ angles, and absolute error in δ and γ trajectories between the Marker and the Proposed algorithms. (a,c,e) Data related to the best performance test ($P2_3$). (b,d,f) Data related to the worst performance test ($P1_2$).

3.4. Conclusion

In this chapter, a depth-based method for joint estimation in side view has been described. This approach can be used to extract parameters for the objective evaluation of a mobility assessment test that requires to stand up from a chair and walk for a short distance. The system is unobtrusive since it does not require to wear any device or physical marker, and the main novelty is given by the operation on the sagittal plane. The comparison with other markerless estimation algorithms, provided in Microsoft and OpenNI SDKs, resulted in better performance in terms of error in the absolute position and also in terms of the estimation of fundamental angles during the sit-to-stand movement.

The performance in the estimation of torso and knee angles has been evaluated considering a marker-based stereometric system as a reference. The computation of the error has required a synchronization algorithm to associate the data captured by different systems: the Kinect sensor and the stereometric system. In particular, it has been necessary to compensate the different starting times and the different sampling rates of the two capturing systems.

The proposed approach could be adopted in cases for which it is not possible to place the sensor in front of the human, due to limitations of the environment, or when it is required to perform the analysis on the sagittal plane.

Chapter 4.

Depth and IMU based algorithm for mobility assessment and fall detection

Mobility assessment tools, such as Timed Up and Go (TUG) test have been instrumented with different sensors and technologies, mainly vision-based devices or wearable sensors. However, the joint use of different technologies may help to overcome the limitations of each one. The main problem arising from the use of acceleration data from wearable sensors is the the lack of information about the context, which may increase the number of false positives. On the other hand, vision-based sensors provide a considerable amount of information but it is difficult to extract features from raw data. A complication arising from the joint use of different devices is the development of synchronization algorithms that have to be used to correctly associate samples from different sensors.

The setup of the proposed system for TUG is shown in Figure 4.1, and it has been chosen beacuse it can be easily implemented in a home environment, assuming to have enough space. The wearable sensor is placed on the chest of the subject ready to perform the test, while the RGB-D device is placed in front of the chair, at a distance of about 3 meters from it, with an height of 1.5 meters from the floor. The sensors are Kinect v2 and an IMU by SHIMMER Research [128]. The SHIMMER sensor v1.3 is the one used in this work, and the main components of the IMU are:

- CPU: Texas Instruments MSP430F1611, 8 MHz clock, 8 channels A/D (12 bit);
- accelerometer: Freescale MMA7260Q, 3 axis, range: $\pm1.5/2/4/6$ g;
- communication interface: Chipcon CC2420 IEEE 802.15.4 Transceiver, BT class 2.

SHIMMER firmware runs on TinyOS, which is an event-driven operating system designed for sensor network nodes with limited resources. The firmware can be

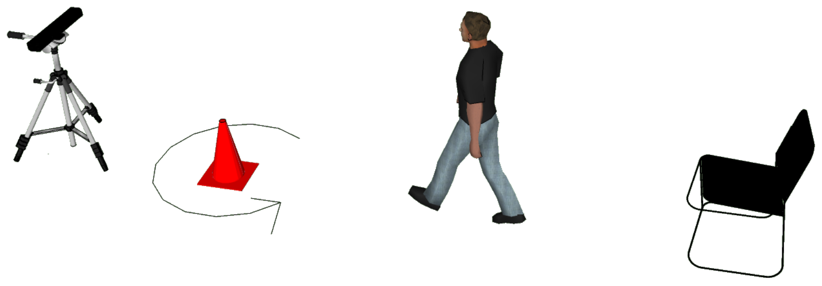


Figure 4.1.: Setup for Timed Up and Go test.

modified to define the structure of the packet where data are encapsulated or how the data should be delivered to the PC.

Fall detection is another application that could benefit from the joint use of wearable and vision-based technologies. The same sensors and configurations presented for TUG could be used for fall detection, allowing to design an integrated system that could be installed in home environment. It can be set as working in fall detection modality, and switched to the TUG modality when required. Fall detection solutions using RGB-D sensors have been deeply investigated by other researchers in literature. Considering RGB-D devices, many approaches are based on the processing of raw depth data [129, 130, 131, 132, 133], whilst others exploit multiple information, such as skeleton or RGB images [134, 135, 136, 137, 138]. Considering the chosen setup, with the availability of skeleton data and the fusion with acceleration samples, it is possible to design simple and effective fall detection algorithms, able to discern between different types of falls and some Activities of Daily Living (ADLs). It is worth clarifying that the use of the term ADL in this thesis does not have to be intended in a clinical sense. The idea is to collect a dataset to be used to test the fall detection algorithms and to label as an ADL any action/activity that is very common in the selected scenario and for which the movements can be similar to a fall.

The integrated system for fall detection and for the mobility assessment has been described in [139]. The synchronization approach and the algorithms to extract parameters from the TUG test are detailed in [140] whilst the fall detection algorithms

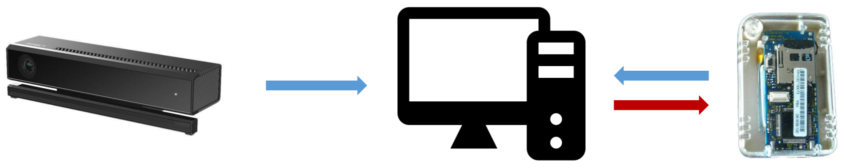


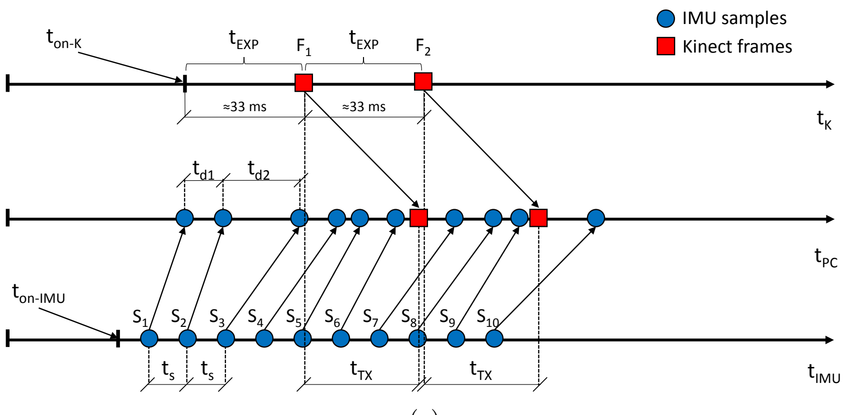
Figure 4.2.: System configuration with Kinect v2 and SHIMMER IMU connected to the same PC through USB and Bluetooth connection.

have been presented in [141].

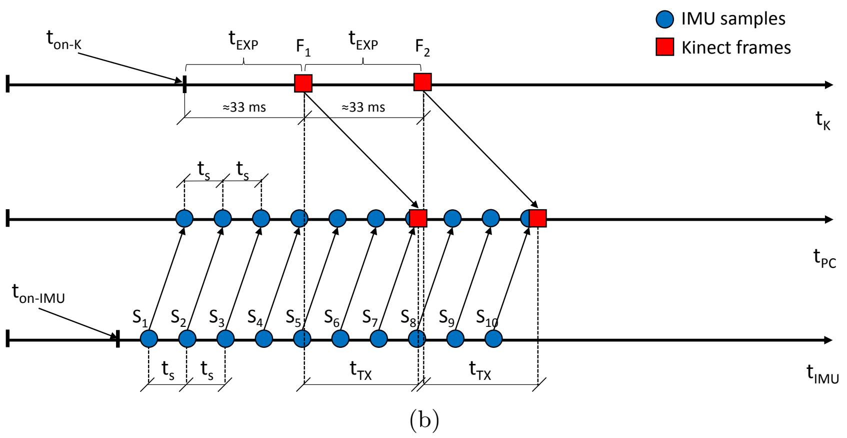
4.1. Synchronization algorithm

As briefly introduced, a synchronization procedure has to be implemented to correctly associate samples generated asynchronously from different sources, and to be sure that events captured by different sensors refer to the same observed phenomenon. The system configuration consists in a RGB-D sensor connected via USB port to a PC and an IMU linked via Bluetooth to the same machine. As can be noticed in Figure 4.2, the communication between Kinect and the PC is unidirectional because the data retrieval can only be activated or deactivated using APIs while the programmable firmware running on the IMU enables a bidirectional communication.

The synchronization issue between the Microsoft Kinect and a wearable sensor is shown in Figure 4.3(a), where the time axes of the three devices involved in the data acquisition process are shown: time axis of Kinect (t_K), PC (t_{PC}) and Inertial Measurement Unit (t_{IMU}). Data are affected by a delay between the time of their capture and the time of arrival at the PC and different delay contributions are involved in the process of acquisition and transmission of data from the sensors to PC. A synchronization process simply based on the time of arrival at the PC may be not accurate. The transmission time of Bluetooth packets is influenced by the channel conditions. The acceleration samples, which are captured with a fixed sampling time (t_s), are usually separated by different delay times when received at the PC (t_{d1}, t_{d2},...). The problem of synchronizing devices that are connected via Bluetooth (the PC and the IMU node in this case) has been addressed previously by Wåhslén *et al.* [142] using linear regression, and the result is visible in Figure 4.3(b), where the delay between consecutive samples from IMU is equal to the sampling time (t_s). The synchronization of Kinect and PC has to follow a different approach



(a)

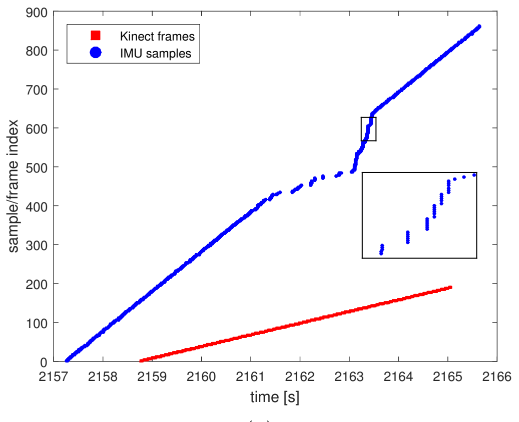


(b)

Figure 4.3.: (a) Delay times of Kinect and wearable device connected to a PC; (b) same delay times after linear regression.

because it is not possible to have access to the sensor internal clock, and to obtain a timestamp related to the acquisition of each frame. The total delay time of a Kinect frame is made up two contributions: the exposure time (t_{EXP}), which is the time required to acquire data, and the transmission time (t_{TX}), which is needed to encapsulate and to send data to the PC. Once these times are computed and known, they can be exploited to synchronize data from the RGB-D sensor to the PC, and consequently to the IMU.

Considering the situation shown in Figure 4.3, if a Kinect frame and an acceleration sample would be associated considering only the time of arrival, without any synchronization procedure, the frame F_1 would correspond to the sample S_6 and the frame F_2 to S_9. The implementation of a synchronization algorithm should correctly



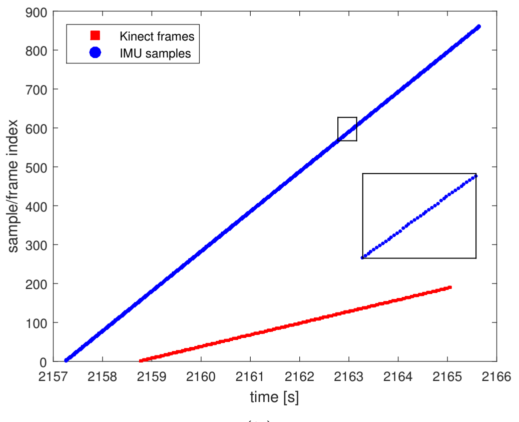


Figure 4.4.: (a) Time of arrival at the PC for Kinect frames and IMU samples, correlated to the their indices; (b) same delay times after linear regression performed on the timestamps of IMU samples.

associate the frame F_1 to the samples S_3-S_4, and the frame F_2 to S_6-S_7.

4.1.1. IMU packets synchronization

The synchronization of IMU samples with the PC is based on linear regression, since the only problem with this type of data is related to the Bluetooth communication that sometimes does not deliver packets with the same rate, but it may send bursts of data with a very low delay and no data for another time. This phenomenon can be noticed in Figure 4.4(a), where the x-axis of the plot is represented by the timestamp taken when the packet from IMU or the frame from Kinect arrive at the PC, and y-axis is the number of the sample. The amount of data received from the IMU is much greater than the number of frames captured by Kinect, because the sampling rate of the Shimmer is 10 ms while Kinect outputs data approximately every 33 ms. Using a linear regression algorithm, the curve related to the wearable device is linearized, obtaining the result shown in Figure 4.4(b). The zoomed area shows that delays between subsequent samples have been corrected.

This is the only delay regarding the IMU device that should be compensated because the amount of time needed by the Shimmer node to read data samples from the accelerometer and to perform A/D conversion is four clock cycles. IMU is equipped with a 32.768 kHz crystal, which means that an acceleration sample can be acquired with less than 0.13 milliseconds. Considering a sampling frequency of

100 Hz, the ratio between the sampling time and the acquisition time is more than 50. When sampling a frame from a camera, the acquisition time is quite different, thus transmission and exposure times of Kinect have to be accurately estimated.

4.1.2. Kinect delay estimation

The delay times estimation for the frames captured by Kinect has been performed by using an Arduino board, connected to the same PC as the Kinect camera, which controls some LEDs (7 in this case). The round-trip time between the PC and the Arduino serial interface has been estimated over 1500 transmissions. In the implemented setup, an average value of 3.2 milliseconds has been obtained. Different procedures have been developed to estimate the transmission and exposure times for Kinect. Since it is not possible to have a timestamp from the internal clock, data timestamping is performed when the frame arrive to the PC using C++ functions *QueryPerformanceCounter* (QPC) [143] and *QueryPerformanceFrequency* (QPF) [144]. The former one counts the number of ticks of the performance counter from the system initialization with high resolution (less than 1 us) while the QPF retrieves the frequency of the performance counter.

Considering the transmission time for the RGB frames, the implemented procedure follows the steps detailed below, and represented in Figure 4.5(a):

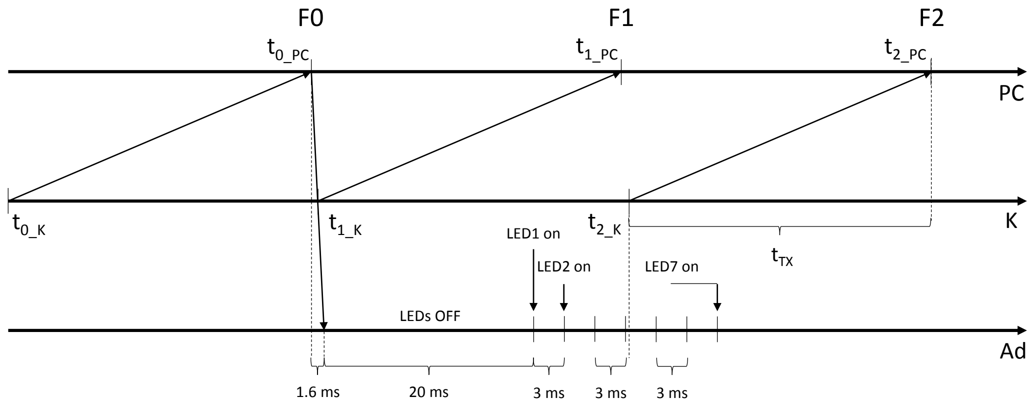
1. When the PC receives the frame F0 from Kinect, it sets a timestamp (t_{0_PC}) and sends a predefined command to the Arduino board.

2. The Arduino board receives the command, set a delay of 20 milliseconds, and switches on one LED at a time with a delay of 3 ms.

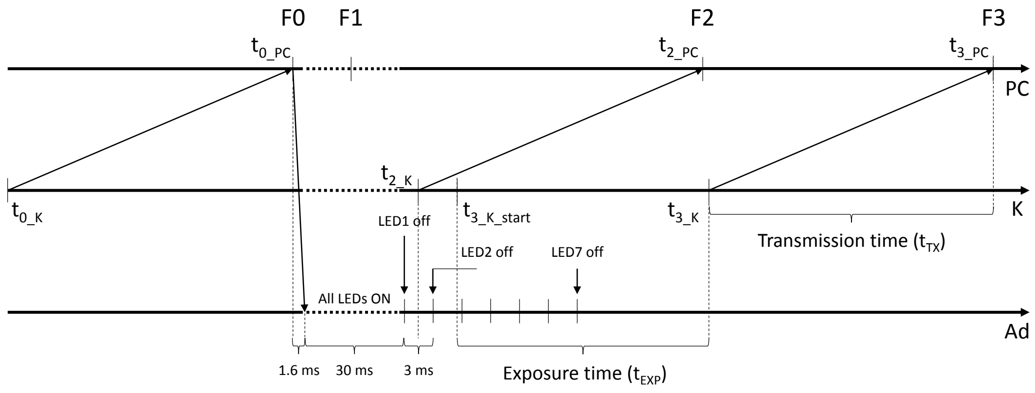
3. When frame F2 arrives at the PC, the timestamp (t_{2_PC}) is taken and the number of LEDs that are ON in that frame gives an estimation of the time t_{2_K} with an error of 1.5 ms.

4. The transmission time for the frame F2 can be computed from the difference between t_{2_PC} and t_{2_K}.

The same procedure, increasing only the delay to 35 ms, has been implemented to evaluate the transmission time of the IR frame. For this stream, IR LEDs operating at 830 nm wavelength are used. Those LEDs cannot be detected in the depth frame of Kinect v2. For the evaluation of this stream, a moving object in the Kinect coverage area can be used to estimate additional delays between IR and depth frames



(a)



(b)

Figure 4.5.: Evaluation scheme of the RGB transmission time (t_{TX}) (a) and RGB exposure time (t_{EXP}) (b) for Kinect v2.

(depth is obtained from IR data). The time delay between frames showing the same displacement is measured. This approach has been applied also to evaluate the transmission times of RGB, IR and depth frames for Kinect v1. In this case the technology used for depth estimation is different (structured light instead of Time of Flight) and the IR LEDs can be detected from the depth frame.

A different procedure, which is composed by the steps detailed below and in Figure 4.5(b), is adopted for the exposure time of the RGB frame of Kinect v2.

1. When the PC receives the frame F0, it sets a timestamp (t_{0_PC}) and sends a predefined command to the Arduino board.

2. The Arduino board switches on all the 7 LEDs when it receives the command

and sets a waiting time of 30 ms; then it starts switching OFF the LEDs (one at a time) with a delay of 3 ms.

3. When frame F3 arrives at the PC, the timestamp t_{3_PC} is taken and the number of LEDs which are ON in F3 gives an estimation of $t_{3\_K\_start}$ with an error of 1.5 ms.

4. Since the time t_{3_K} is given by the difference between t_{3_PC} and the transmission time (t_{TX}), the exposure time is represented by the time delay between t_{3_K} and $t_{3\_K\_start}$.

The delays between the arrival of the command at the Arduino and the actions on the LEDs (30 ms in this last case) have been experimentally chosen to center the on/off leds sequence around t_{2_K} or $t_{3\_K\_start}$ time. The used delays could be removed or at least reduced without increasing the uncertainty using a greater number of LEDs. This approach has been used also for RGB, depth and IR frames of Kinect v1, while a different procedure is implemented for the exposure time of IR frames for Kinect v2, which is much lower than the RGB one. The Arduino board switches ON and OFF 7 IR LEDs with a delay of 1 ms. The exposure time can be estimated by counting the number of LEDs that appear ON in a single frame.

A laptop equipped with Intel core 2 Duo @ 2.53 GHz, 4 GB RAM, Windows 7 operating system has been used to capture data from Kinect v1. A number of 200 frames has been captured for each stream. For Kinect v2, 75 frames are captured using a desktop PC equipped with Intel i7 @ 3.5 GHz, 16 GB RAM, Windows 8.1 operating system. The results in terms of exposure and transmission times, for each available data stream, are shown in Table 4.1. Considering the results for Kinect v1, the transmission time for the depth frame is higher than the other two data streams. The main reason could be the time required to extract the depth information from the IR frame, by using the structured light technique. The exposure time is approximately the same for all the three streams and it is quite high, since it is around 30 ms. Regarding Kinect v2, Table 4.1 shows that the transmission time for the RGB frames is much greater than the transmission time for IR and depth frames (almost two times). The reason may be the difference in the amount of data that has to be delivered: the resolution is 1920×1080 for RGB and 512×424 pixels for IR and depth. The RGB exposure time of Kinect v2 is comparable to the previous version, while the IR and depth exposure time are much lower.

The delay times for skeleton stream available with Kinect have not been included

Table 4.1.: Kinect v1 and v2 exposure and transmission times.

Data stream	Exposure time [ms]	Transmission time [ms]
Kinect v1		
RGB	28.4 ± 2.0	15.0 ± 2.2
IR	31.3 ± 2.2	16.4 ± 2.0
Depth	29.1 ± 6.3	28.2 ± 5.7
Kinect v2		
RGB	28.5 ± 1.2	31.5 ± 1.1
IR	3.0 ± 1.2	16.0 ± 1.0
Depth	3.0 ± 1.2	18.5 ± 1.0

in the results because, since the skeleton joints are extracted from a depth frame, the Microsoft SDK provides the information necessary to associate a skeleton frame to a depth frame.

4.1.3. Synchronization results

The synchronization procedure between Kinect and the IMU node consists of some steps, detailed below.

1. The linear regression algorithm is applied to the packets received by the PC from the IMU. This approach is able to compensate the channel effect of the Bluetooth and to have the same delay between all the subsequent packets.
2. Computation of the Kinect offset, by adding the transmission time to a half of exposure time for the frame of interest.
3. Subtraction of the offset to the timestamp taken at the time of arrival at PC.
4. The IMU packets closest to each frame are determined and associated to it.

The accuracy of the transmission and exposure times estimation, and the results obtained from the proposed synchronization approach, has to be verified. The IMU device (SHIMMER) has three programmable LEDs with three colors: red, orange, yellow. The LEDs can be switched ON and OFF according to some events and those events can be captured by the Kinect camera.

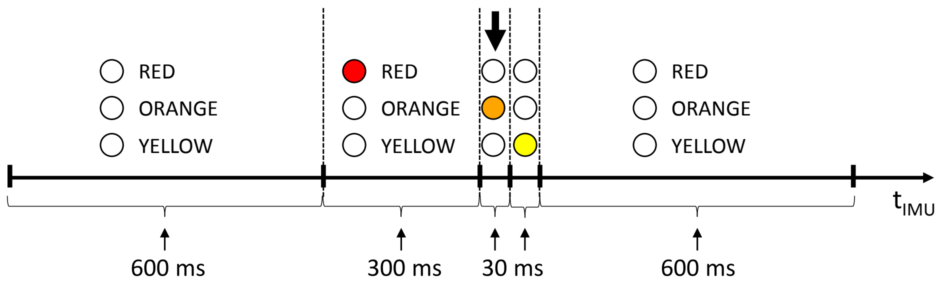
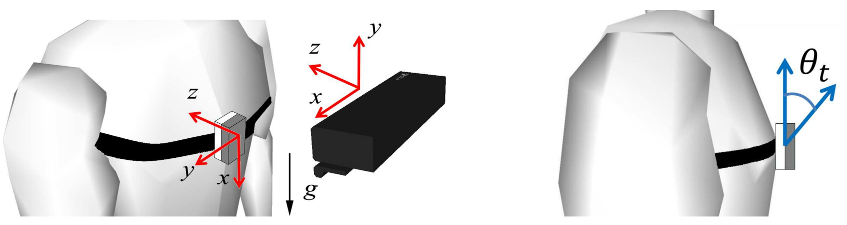


Figure 4.6.: SHIMMER LED pattern implemented to verify the accuracy of the synchronization algorithm.



(a) (b)

Figure 4.7.: (a) Orientation of IMU and Kinect v2 for TUG test, (b) torso's angle θ_t computed from inertial data.

The LEDs on the SHIMMER device are switched ON and OFF according to the sequence shown in Figure 4.6. The node switches ON the orange LED and sends a predefined packet to the PC. This packet is received and the synchronization algorithm is implemented, to determine which is the closest RGB frame received by the PC from Kinect. The proper RGB frame should display the Shimmer device with its orange LED ON. In order to have the lowest uncertainty, the ON event is kept only for a time window of 30 ms.

This evaluation procedure has been repeated for 48 frames captured by Kinect v2 and the correct frame has been detected in 45 out of the 48 events (93.8%). In the other cases, the maximum estimation error is one frame only. Considering the Kinect v1 sensor, a percentage of correct estimation of 99.4% has been achieved over 180 trials. As for Kinect v2, the maximum error is one frame.

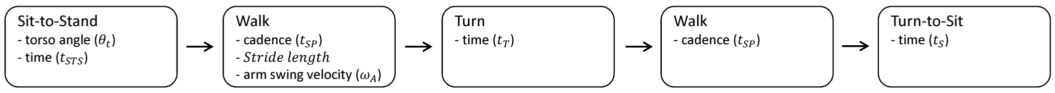
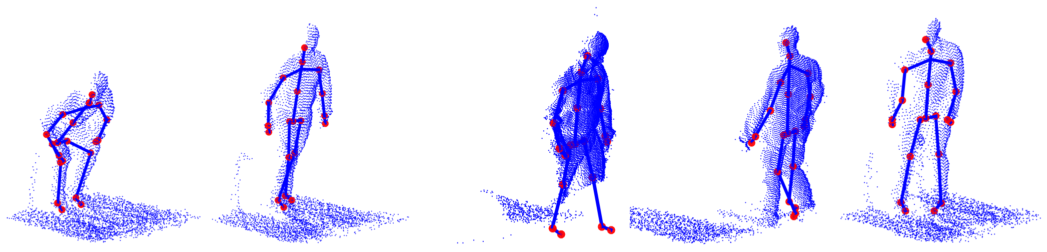


Figure 4.8.: Scheme of TUG test, constituted by 5 phases. Different parameters are computed during each phase.



(a) (b) (c) (d) (e)

Figure 4.9.: Point cloud and skeleton of a subject performing the TUG test.

4.2. Parameters estimation from TUG

In the setup implemented for TUG test, the sensors have been positioned as previously introduced, with the IMU attached on the chest of the person and the Kinect placed at a distance of 3 meters. The relative orientations are defined by Figure 4.7, where it can be noticed that the x-axis of the IMU is oriented according to the gravity acceleration vector. Thus, the torso's angle θ_t can be computed as the displacement with respect to the x-axis of the IMU. The TUG test can be divided in 5 different phases: Sit-to-Stand (STS), Walk, Turn, Walk, Turn-to-Sit. Those phases are summarized in a scheme in Figure 4.8, where the indices evaluated for each phase are also detailed. Figure 4.9 shows the point clouds and the skeletons of a person during the 5 phases.

STS is the first phase where the person, who is sitting on a chair at the beginning, stands up to start walking (see Figure 4.9(a)). In this first phase, two indices are evaluated: the maximum inclination of the torso angle (θ_t) and the time duration of the movement (t_{STS}). The torso's angle is computed from the raw acceleration data (x_{raw}, y_{raw}, z_{raw}) provided by the IMU. Raw data have to be converted into gravity accelerations (X, Y, Z) taking into account possible biases (x_{bias}, y_{bias}, z_{bias}), which are identified by testing the $1g$ orientation for each axis. By defining as x_{0g} the raw acceleration for x-axis in $0g$ configuration, and equivalently y_{0g}, z_{0g} for y and z axes, and considering $V_{sens} = 300mV/g$ as the accelerometer sensitivity when it is set in

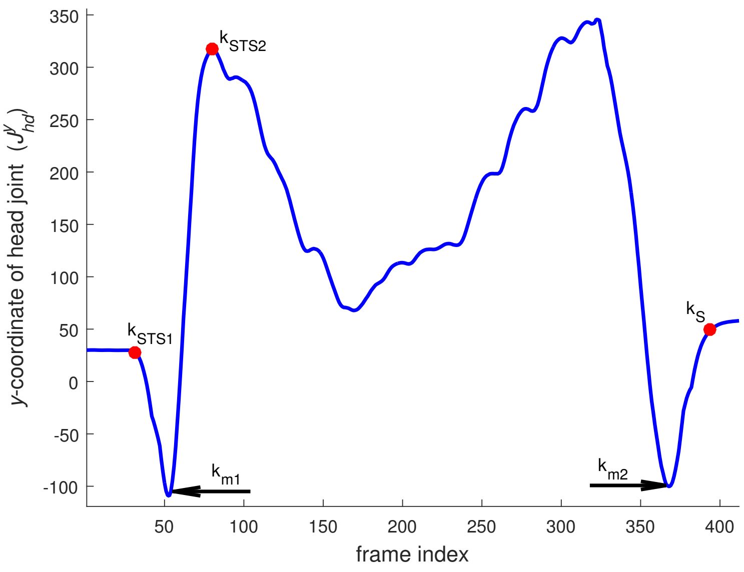


Figure 4.10.: y-coordinate of head joint in a complete TUG test: J^y_{hd}.

the range $\pm 4g$ [145], the equations used to obtain the gravity accelerations starting from raw data are defined in (4.1):

$$X = \frac{x_{raw} - x_{0g} + x_{bias}}{V_{sens}}, \ Y = \frac{y_{raw} - y_{0g} + y_{bias}}{V_{sens}}, \ Z = \frac{z_{raw} - z_{0g} + z_{bias}}{V_{sens}} \tag{4.1}$$

Considering the obtained X, Y and Z, the θ_t angle (in degrees) is given by (4.2):

$$\theta_t = \max\left\{90 - \tan^{-1}\left(\frac{X}{\sqrt{Y^2 + Z^2}}\right)\right\} \tag{4.2}$$

The extraction of timing parameters during Sit-to-Stand (t_{STS}) is performed considering the y-coordinate of the head joint (J^y_{hd}), which is shown in Figure 4.10 for one test sample. The idea is to evaluate the time interval between the frame (k_{STS1}), where the subject starts the movement, and the frame (k_{STS2}) where the person has finished the STS phase and starts the walking phase. If a TUG execution is composed by K skeleton frames, the k_{STS2} frame is the one where the J^y_{hd} assumes the maximum value, in the first half of the test, satisfying the condition (4.3):

$$J^y_{hd}(k_{STS2}) = \max\left\{J^y_{hd}(k)\right\}, \quad k = 1, 2, \ldots, K/2 \tag{4.3}$$

With the same reasoning, the k_{m1} frame is the one showing the minimum coordinate

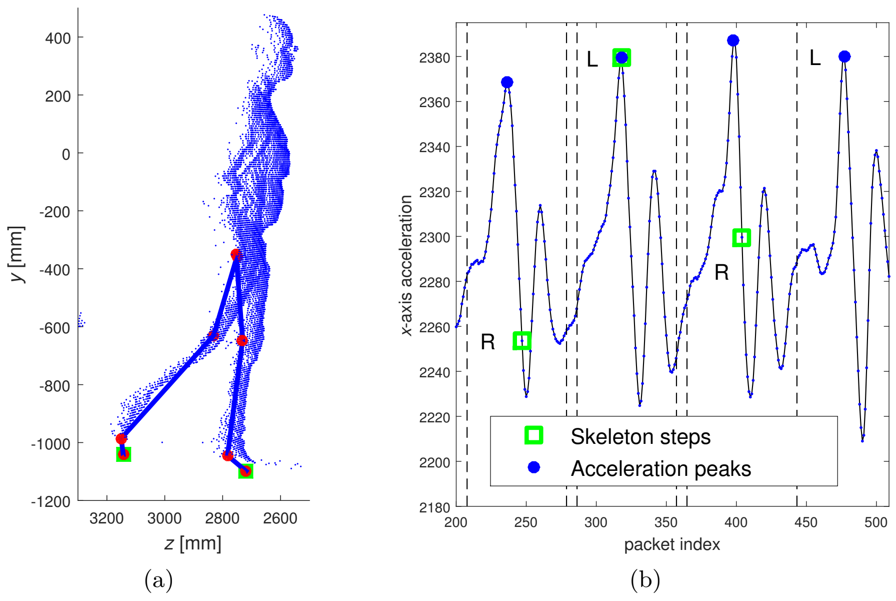


Figure 4.11.: (a) Point cloud of the subject during the Walk phase of the test with skeleton of lower body part. (b) x-axis acceleration data with skeleton steps and acceleration peaks.

for head joint in the STS phase:

$$J_{hd}^{y}(k_{m1}) = \min\left\{J_{hd}^{y}(k)\right\}, \quad k = 1, 2, \ldots, k_{STS2} \tag{4.4}$$

and k_{STS1} is defined as the first frame in the frame interval $[2, k_{m1}]$ satisfying the condition (4.5):

$$k_{STS1} = \min\left\{k\right\} \quad | \quad |J_{hd}^{y}(k) - J_{hd}^{y}(k-1)| > Th_{1\%}, \quad k = 2, \ldots, k_{m1} \tag{4.5}$$

where $Th_{1\%} = |J_{hd}^{y}(1) - J_{hd}^{y}(k_{m1})|/100$. By defining as Δ_t the time difference between two consecutive skeleton frames (33 ms), the equation (4.6) gives the time required to complete the STS phase:

$$t_{STS} = (k_{STS2} - k_{STS1})\Delta_t \tag{4.6}$$

During the first Walk phase, represented by the point cloud and skeleton in Figure 4.9(b), data from Kinect and from IMU are jointly considered to extract the cadence t_{SP} and the *Stride length*, while some skeleton joints are involved in the process of computation of the arm swing velocity ω_A. At the beginning, an approximate identification of the steps is carried out using skeleton information, as shown in Figure 4.11. In particular, Figure 4.11(a) shows the point cloud of the subject during the Walk phase, where the steps are initially identified looking for the maximums in the distance among the feet along the Kinect z-coordinate. Using the synchronization algorithm, skeleton steps can be related to raw acceleration samples in x-axis, as in Figure 4.11(b). The average time distance of the skeleton steps defines the width of some search intervals, represented by vertical dashed lines. An algorithm for accurate step identification is implemented within each interval and the time instants of steps are associated to the most important acceleration peaks. The algorithm can also consider some intervals where the skeleton step is not identified, such as the fourth one in Figure 4.11(b). This is due to the fact that the lower limbs are out of the coverage area of Kinect when the subject is approaching the Turn phase and the skeleton estimation of lower joints is not reliable (see Figure 4.9(c)). The acceleration peaks corresponding to steps not detected by the skeleton have to satisfy amplitude and time properties similar to the previously identified ones, to avoid the identification of false events. Each acceleration peak, corresponding to a step event, is associated to the corresponding foot. The skeleton position is exploited to detect which is the closest foot to the sensor. For example, the label R to the first skeleton peak in Figure 4.11(b) means that the step is done with the right foot. When skeleton data is not reliable, such as for the fourth step in Figure 4.11(b), the algorithm associates the step to the opposite foot of the previous one. Information about cadence is extracted also during the second Walk phase, i.e. when the person walks towards the chair. In this phase, as visible in Figure 4.9(d), the skeleton data are not reliable and the steps are detected by exploiting only the wearable sensor. In this operation, time and amplitude properties learned during the first Walk phase are exploited. For the same reason, the *Stride length* is computed only in the first Walk phase.

The angular velocity of arm-swing is estimated within the time intervals defined by the steps. The angle composed by the wrist and shoulder joints position at k-th frame, and the position of the same wrist at frame $k-1$ is considered. By dividing the values of this angle by the sampling time of the Kinect and by averaging them, it is possible to estimate the angular velocity (ω_A) for each gait cycle. Considering the position of the shoulder joint in the k-th skeleton frame ($J_{sh}(k)$), and $J_w(k)$ the same for the wrist joint, two difference vectors can be defined:

$$d_{sw}(k) = J_{sh}(k) - J_w(k) \tag{4.7}$$

$$d_{sw}(k-1) = J_{sh}(k) - J_w(k-1) \tag{4.8}$$

The angular velocity of the arm $\omega_A(k)$ at k-th frame is:

$$\omega_A(k) = \frac{1}{\Delta_t} \cos^{-1}\left(\frac{d_{sw}(k)}{\|d_{sw}(k)\|} \cdot \frac{d_{sw}(k-1)}{\|d_{sw}(k-1)\|}\right) \tag{4.9}$$

The index of interest is ω_A, which is the average angular velocity over each gait cycle.

The Turn phase refers to the turning movement performed when the subject is approaching the sensor and separating the two walking phases. An important parameter in this phase is the time required to perform the 180° rotation. The shoulders and head joints are used to define an orientation vector which gives the moving direction of the person, and helps to identify the instant of start and stop of the body rotation, giving the turning time (t_T). The events of turning are recognized checking the angle between the orientation vector and a reference direction, which is the Kinect z-axis, since it represents the main walking direction. In the final part of the test, when the subject performs a turning movement to sit down, the frame related to turning and the k_S frame, calculated with the same approach of k_{STS1} in equation (4.5), are used to compute the time duration of the sit phase (t_S). The k_S frame, as can be noticed in the J^y_{hd} curve in Figure 4.10, is the final frame of the test and can be used also to extract the total time duration of the TUG test (t_{TUG}), considering k_{STS1} as the start frame.

Different tests have been performed in a laboratory environment. The only parameter related to the environment that should be controlled is represented by the distance between Kinect and the chair, no constraints have been considered for the background. A number of 20 healthy subjects, aged between 22 and 39, with different build and height, have been recruited for the tests. Each person repeated three times the TUG, thus a total number of 60 executions constitute the TST TUG dataset [146] that has been released to encourage other researchers to work on this topic. The dataset has been collected with a capture software for Kinect v2 named *Complete viewer* and detailed in Chapter A, and it provides the following data:

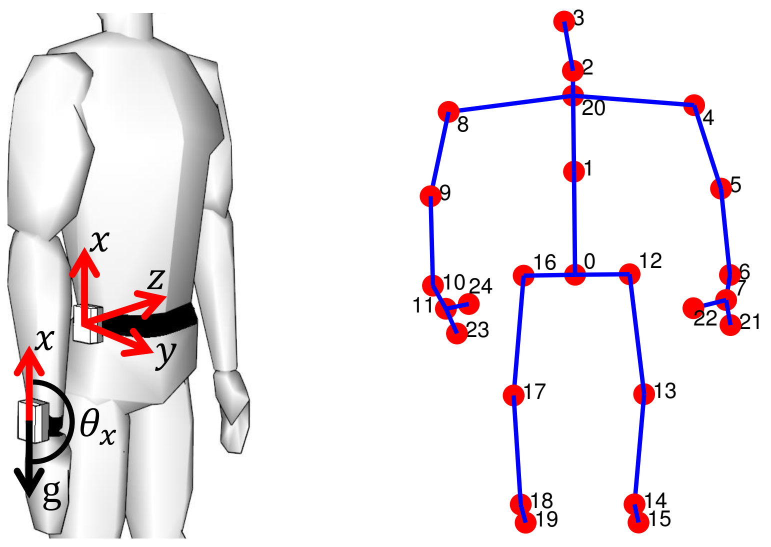
- depth frames from Kinect v2;

Table 4.2.: Average values and confidence intervals for TUG indices.

Index		Average value	Minimum	Maximum
θ_t	[deg]	45.3 ± 13.53	18.0	64.0
t_{TUG}	[s]	9.99 ± 1.88	7.00	13.73
t_{STS}	[s]	1.47 ± 0.41	0.96	2.41
t_S	[s]	2.42 ± 0.54	1.42	3.23
t_T	[s]	1.54 ± 0.59	0.66	3.46
t_{SP}	[s]	1.38 ± 0.21	1.12	2.37
Stride length	[mm]	956 ± 147	743	1295
ω_A	[deg/s]	107 ± 21	67	158

- skeleton joints in depth and world coordinates;
- raw acceleration samples provided by the IMU attached to the chest;
- timing information for synchronization.

The results obtained over the entire test set are summarized in Table 4.2. All the tests have been performed by healthy people, thus the obtained values, especially the ones related to time, show limited standard deviations. The maximum inclination of the torso angle during the STS (θ_t) is 45 degrees in average, with a standard deviation of 14 degrees. However, the difference between the minimum value (18°) and the maximum (64°) is quite high, denoting a different way to perform the movement. The average time required to perform the entire test is about 10 s, where about 1.5 seconds are required by the STS phase (t_{STS}). The t_S time is almost 2.5 s on average, and it is larger than t_{STS} because the former includes also the turning movement. The first turning movement, measured by t_T, lasts 1.54 s on average. The indices related to gait refer to the first gait cycle. In particular, the *Stride length* is about 956 mm, and an average time of 1.38 s is required to complete the cycle (t_{SP}). The angular velocity ω_A is also related to the first gait cycle, considering the left arm swing.



(a) (b)

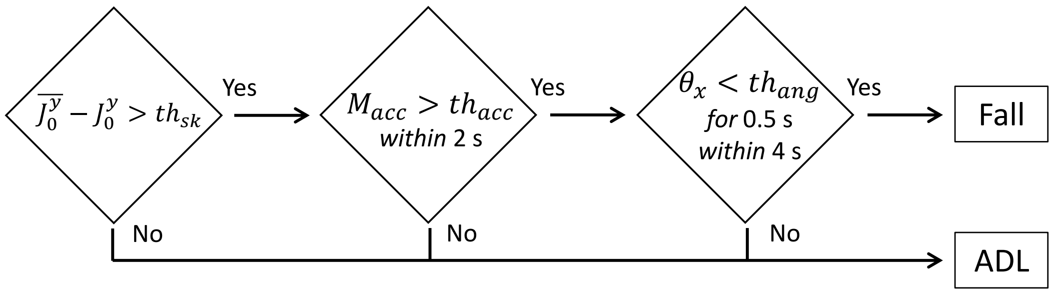
Figure 4.12.: (a) Positions of IMUs on the subject's body. The orientation of x-axis with respect to the gravity acceleration vector is given by the angle θ_x. (b) Skeleton joints estimated by SDK 2.0 of Kinect v2.

4.3. Fall detection

Different parameters can be extracted from RGB-D and acceleration sensors, and different algorithms for fall detection have been implemented and evaluated. The configuration of the Kinect v2 sensor is the same as the TUG application, it is placed at an height of about 150 cm from the floor, monitoring a room, or part of it. Two positions have been investigated for the IMU, which can be attached on the belt and on the wrist, as shown in Figure 4.12(a).

4.3.1. Algorithms for fall detection

A fall detection algorithm is based on acceleration data from the wrist IMU and skeleton joints from Kinect. In particular, as sketched in Figure 4.13, the fall or ADL classification procedure consists of three steps and considers the variation in the skeleton joint position, a peak of acceleration and the orientation angle computed by the wrist accelerometer. The first evaluated parameter is the variation in the vertical coordinate of a skeleton joint, which is the $SPINE\_BASE$ joint (J_0), located at the base of the spine, as represented in Figure 4.12(b). Its y-coordinate (J_0^y) is



$\overline{J_0^y} - J_0^y > th_{sk}$
Yes
$M_{acc} > th_{acc}$ within 2 s
Yes
$\theta_x < th_{ang}$ for 0.5 s within 4 s
Yes
Fall
No
No
No
ADL

Figure 4.13.: Scheme of Algorithms 1 and 2 for fall detection.

considered to calculate a reference value ($\overline{J_0^y}$) in the first captured frames, assuming that the person is standing in front of the sensor. Every new skeleton frame, the difference between the actual value and the reference one is computed and, if that difference exceeds a threshold of 50 cm (th_{sk}), an irregular activity is detected and a timestamp (t_{ir}) is set. If no irregular activity is identified, the algorithm does not consider acceleration data and immediately classifies the movement as an ADL. If an irregular activity is detected, acceleration data extracted from IMU are firstly used to compute the magnitude of acceleration M_{acc} captured by the IMU, defined as:

$$M_{acc} = \sqrt{X^2 + Y^2 + Z^2} \tag{4.10}$$

where X, Y and Z are the unbiased acceleration components along the x, y and z axes of the accelerometer, as defined in (4.1). Kangas et al. [147] suggested that an acceleration of $3g$ may be associated to a fall. An acceleration peak greater than th_{acc}, which is set to $3g$, has to be present within a time interval of 2 seconds centered in t_{ir}. Before taking a decision, the algorithm implements a last check, considering the orientation of the sensor when the event is ended. From Figure 4.12(a) can be derived that, if the person is lying, maybe after a fall, the IMU orientation angle (θ_x) should be close to 90°. This condition should be verified for a not negligible amount of time to reveal a fall. In order to correctly measure the device orientation, the component of acceleration related to body motion has to be removed from raw data. As stated in [148], the daily activities feature a frequency range between 0.3 and 3.5 Hz; thus, to extract only the gravity acceleration component, X, Y and Z have been filtered by a third order Butterworth low pass filter with cut-off frequency of 0.5 Hz. The filter cuts all the frequencies generated by the body motion, leading to a correct estimation of the device orientation. The IMU orientation angle θ_x is

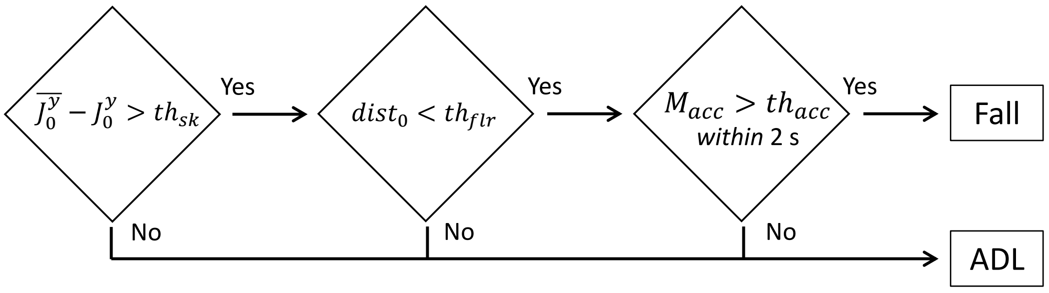

$\overline{J_0^y} - J_0^y > th_{sk}$
Yes
$dist_0 < th_{flr}$
Yes
$M_{acc} > th_{acc}$ within 2 s
Yes
Fall
No
No
No
ADL

Figure 4.14.: Scheme of Algorithm 3 for fall detection.

therefore computed using the following equation [12]:

$$\theta_x = atan2\left(\sqrt{Z^2 + Y^2}, X\right) \tag{4.11}$$

A guard interval should be added to the orientation value representing a person on the floor. Thus, the condition that should be verified is set as a threshold th_{ang} of 90° with a guard interval of 20° for a duration of at least 0.5 s within a time interval of 4 s after t_{ir}. If all the above conditions, summarized in Figure 4.13, are satisfied, the action is classified as a fall; otherwise, if a single condition is not satisfied, the sequence of frames is recognized as an ADL.

The second implemented algorithm works on data captured by the IMU attached on the waist of the person, and it implements the same conditions as Algorithm 1, with the same thresholds. Therefore, it can be summarized by Figure 4.13 as well.

Figure 4.14 represents the scheme implemented in Algorithm 3, where the first condition is, again, a variation in the coordinate of the spine skeleton joint: J_0. The main difference is constituted by the second condition, which requires the evaluation of distance of that joint from the floor. The floor can be calculated from the first available skeleton frame, when the person is standing in front pose, and it is modeled as a plane, described by the following equation:

$$ax + by + cz + d = 0 \tag{4.12}$$

where the constant term d is computed using (4.13):

$$d = -(ax_0 + by_0 + cz_0) \tag{4.13}$$

considering $v_n = [a, b, c]$ as the orthogonal vector to the plane, and $P_0 = [x_0, y_0, z_0]$

as one point belonging to the plane. The vector representing the spine of the person can be seen as the orthogonal vector to the plane, which is v_n. This quantity can be obtained from joints J_0 and J_1 in Figure 4.12 with the following equation:

$$v_n = \frac{J_1 - J_0}{||J_1 - J_0||} \tag{4.14}$$

and one of the ankle joints of the subject (number 14 or 18 in Figure 4.12(b)) can be considered as a point of the floor plane. The equation (4.15) can be used to derive $dist_0$, which is the distance of interest, between the joint J_0 and the floor plane:

$$dist_0 = \frac{|v_n \cdot J_0 + d|}{||v_n||} \tag{4.15}$$

A threshold of 20 cm (th_{flr}) is used to recognize if the person is on the floor. The last condition that Algorithm 3 implements is about the acceleration revealed by the waist accelerometer, as shown in Figure 4.14. A value of M_{acc} greater than $3g$ (th_{acc}) has to be found within a time interval of 2 seconds centered in t_{ir}. Even for Algorithm 3, all the conditions about joint variation, floor distance and acceleration have to be satisfied to detect a fall.

The implementation of multiple conditions based on different data brings to a low number of false alarms, which can make the system unreliable. Moreover, the proposal of a step-based fall detection algorithm may allow to detect the starting of a fall, for example considering only the condition about the variation in the skeleton joint position. Then, the fall event should be confirmed by the other conditions to avoid false positives.

4.3.2. Performance analysis

The performance analysis has been carried out following the experimental protocol detailed in Table 4.3, constituted by different ADLs and falls. Regarding ADLs, the most common actions are included in the dataset, i.e. sitting on a chair (*sit*), picking up an object from the floor (*grasp*), walking or lying down. Different types of fall are also considered, among which *EUpSit*, represented by a backward fall with the subject ending up sitting.

The activities have been simulated in a laboratory environment by 11 healthy subjects, aged between 22 and 39, featuring different height (1.62-1.97 m) and build. As for the TUG, each activity has been repeated three times by each subject involved. The whole dataset [146], which has been collected with *Complete viewer* (see Chapter

Table 4.3.: Experimental protocol for fall detection algorithms.

Category	Activity	Description
ADL	*sit*	The subject sits on a chair
	grasp	The subject walks and grasps an object from the floor
	walk	The subject walks back and forth
	lay	The subject lies down on the mattress
Fall	*front*	The subject falls from the front and ends up lying
	back	The subject falls backward and ends up lying
	side	The subject falls to the side and ends up lying
	EUpSit	The subject falls backward and ends up sitting

Table 4.4.: Average results of the fall detection algorithms, in terms of Sensitivity, Specificity and Accuracy.

Algorithm	Sensitivity	Specificity	Accuracy
Algorithm 1	59%	98%	79%
Algorithm 2	79%	100%	90%
Algorithm 3	99%	100%	99%

A) and released to other researchers working in this field, is constituted by 264 sequences and the available data are:

- depth frames captured by Kinect v2;
- skeleton joints in depth and world coordinates;
- raw acceleration data provided by IMUs;
- timing information for synchronization.

Fall detection algorithms have been evaluated in terms of Sensitivity, Specificity and Accuracy over the entire dataset, and the average results are shown in Table 4.4. Algorithm 1 is the most unobtrusive one because it exploits data from the IMU mounted on the wrist, which can be assimilated to a smartwatch. Despite it shows

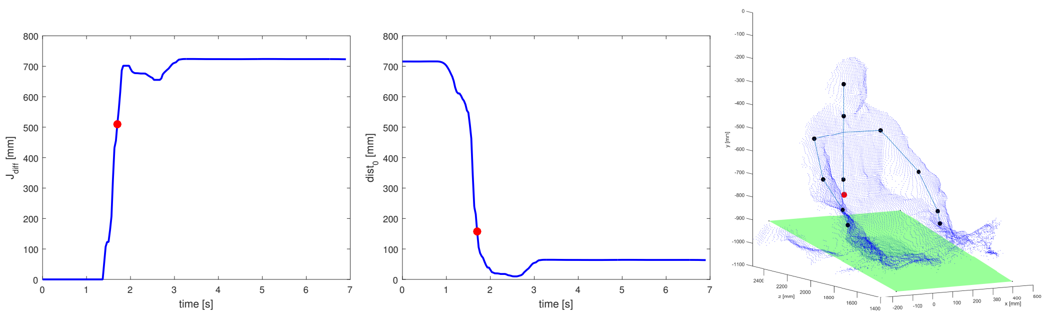
Table 4.5.: Accuracy of the three fall detection algorithms for each activity of the dataset.

Category	Activity	Accuracy		
		Algorithm 1	Algorithm 2	Algorithm 3
ADL	*sit*	97%	100%	100%
	grasp	100%	100%	100%
	walk	100%	100%	100%
	lay	97%	100%	100%
Fall	*front*	54%	100%	97%
	back	82%	100%	100%
	side	48%	100%	100%
	EUpSit	52%	18%	100%
Average		79%	90%	99%

a Specificity of 98%, it has a Sensitivity of 59%, which means that a low percentage of falls is correctly revealed. The main issue of Algorithm 1, which has an average Accuracy of 79%, is given by the θ_x angle giving the orientation of the accelerometer. Even if the subject is on the floor, the arm can have different orientations, which may be similar to the orientations calculated when the person is standing or sitting. Considering the results in Table 4.5, where the accuracy obtained for each class is shown, it can be noticed that Algorithm 1 struggles in particular in the detection of *side* fall, featured by an accuracy of 48%. The highest accuracy among the falls is achieved by the *back* one (82%).

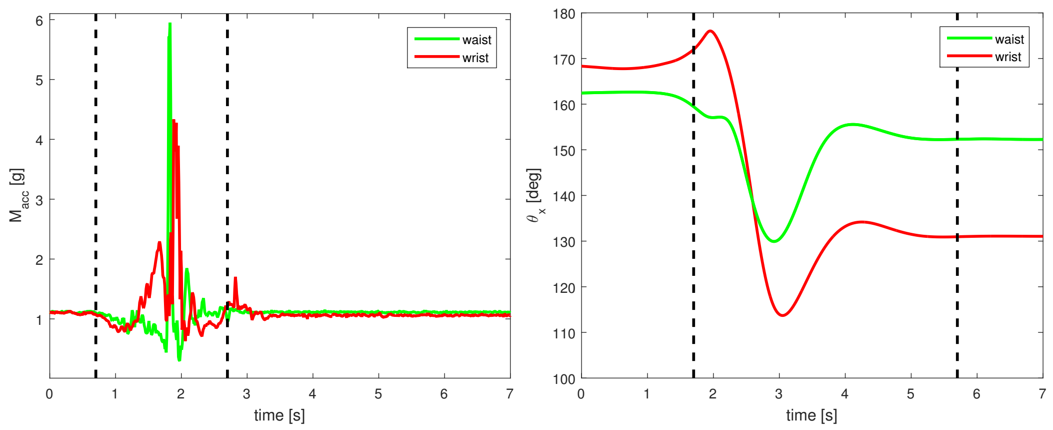
The IMU attached on the waist, even if it is more invasive, provides a reliable information about the orientation of the upper body of the subject. A Sensitivity of 79% is achieved by Algorithm 2, which is featured by an Accuracy of 100% for each activity, except the *EUpSit* fall test. In this particular case, the condition on the orientation of the accelerometer is not verified because the torso remains perpendicular to the floor in almost all the tests.

EUpSit falls are correctly detected with Algorithm 3, that is based on the distance between the spine joint and the floor and does not rely on IMU orientation. This algorithm achieves average Sensitivity and Accuracy of 99%, and a 100% Specificity.



(a) (b) (c)

Figure 4.15.: Curves describing J_{diff} (a) and $dist_0$ (b) during an *EUpSit* fall. Person's point cloud with detected floor plane (c).



(a) (b)

Figure 4.16.: (a) Acceleration and (b) orientation values computed by waist and wrist mounted IMUs during an *EUpSit* fall.

Parameters computed during an *EUpSit* fall are shown in Figures 4.15 and 4.16. The quantity $J_{diff} = \overline{J}_0^y - J_0^y$, which is the variation of the y-coordinate of J_0 axis during the event, is shown in Figure 4.15(a), where t_{ir}, i.e. the time instant indicating an irregular activity, occurs about 1.8 s after the beginning of the action. Considering Algorithm 3, the second condition is about the distance $dist_0$ between the floor and the J_0 joint, which has to be lower than th_{flr}, set at a value of 20 cm, as shown in Figure 4.15(b). The point cloud of the subject after a *EUpSit* fall is shown in Figure 4.15(c), where the joint J_0 is highlighted by a red circle and the green area models the floor plane. The last condition implemented by Algorithm 3 is about the acceleration peak (greater than th_{acc}) revealed by the waist accelerometer in a time window of 2 seconds centered in t_{ir}. As can be noticed in Figure 4.16(a), this condition is verified, and the fall is correctly detected. This dangerous event is not detected by Algorithm 1 and 2 because the accelerometer orientations from both the IMUs do not give an angle lower than the threshold of th_{ang}, even if the guard interval is considered, as can be noticed from Figure 4.16(b).

4.4. Conclusion

In this chapter an integrated system for fall detection and for the mobility assessment based on vision and wearable data fusion has been described. The characterization of delay times of data from Microsoft Kinect has been carried out and a synchronization procedure based on the compensation of delays has been developed. The evaluation of synchronization errors resulted in a correct association of the Kinect frame to the IMU sample with a percentage of 99.4% in Kinect v1 and 93.8% in Kinect v2. The delay times of RGB-D data can be used also in other synchronization scenarios, when it is needed to know exactly when a frame has been captured from the environment.

The implemented mobility assessment test is the Timed Up and Go and takes advantage of the fusion between RGB-D and acceleration information especially in the identification of gait related parameters. The proposed approach can provide more detailed objective indices than other systems that usually provide only the time needed to perform the test. The setup required to execute the test is more flexible than the one described in Chapter 3, and it can be used also at home assuming to comply with the condition on the distance, which is a requirement of the TUG test.

Fall detection algorithms also benefit from the joint usage of multimodal data mainly because, considering the fusion of data from a wearable device on the waist of the subject and the skeleton joint coordinates, it is possible to simplify the data processing. The collection of a dataset including skeleton and depth data from Kinect v2 and acceleration samples from an IMU has allowed the design of algorithms to discern between 4 different types of falls and 4 of the most significant ADLs.

Chapter 5.

Human action recognition

This chapter proposes different solutions for Human Action Recognition using skeleton joints extracted from Kinect. The algorithms are based on 3D joint coordinates and perform classification process using a multi-class SVM. However, different features representation methods have been investigated, providing also experimental results on several RGB-D datasets. As described in Subsection 2.2.3, different datasets may require different evaluation schemes, among which leave-one-actor-out or leave-one-out cross-validation, cross-subject or cross-view tests. For this reason, the comparison of the performance obtained by the proposed algorithms to other approaches in literature is realized under the same testing conditions.

The first proposed solution is based on Activity Feature Vectors (AFV) that are constituted by the most important postures of a sequence of frames. This algorithm has been evaluated on several datasets: KARD [90], CAD-60 [98], UTKinect [85], Florence3D [105], MSR Action3D [68], considering also, for some of them, different subsets with actions related to the AAL scenario.

The second algorithm, named Temporal Pyramid of Key Poses (TPKP), proposes the adoption of the bag of key poses model and the introduction of a temporal pyramid to represent the structure of the action. The TPKP HAR algorithm has been optimized with an evolutionary algorithm obtaining the set of parameters which maximize the performance on MSR Action3D dataset. Finally, this algorithm has been evaluated also on a large-scale dataset recently released: NTU RGB+D [97].

The details of the algorithm named AFV and its performance on five RGB-D datasets have been published in [149]. The TPKP algorithm and the results on MSR Action3D dataset have been presented in [150] whilst its evaluation on NTU RGB+D dataset has been described in [151].

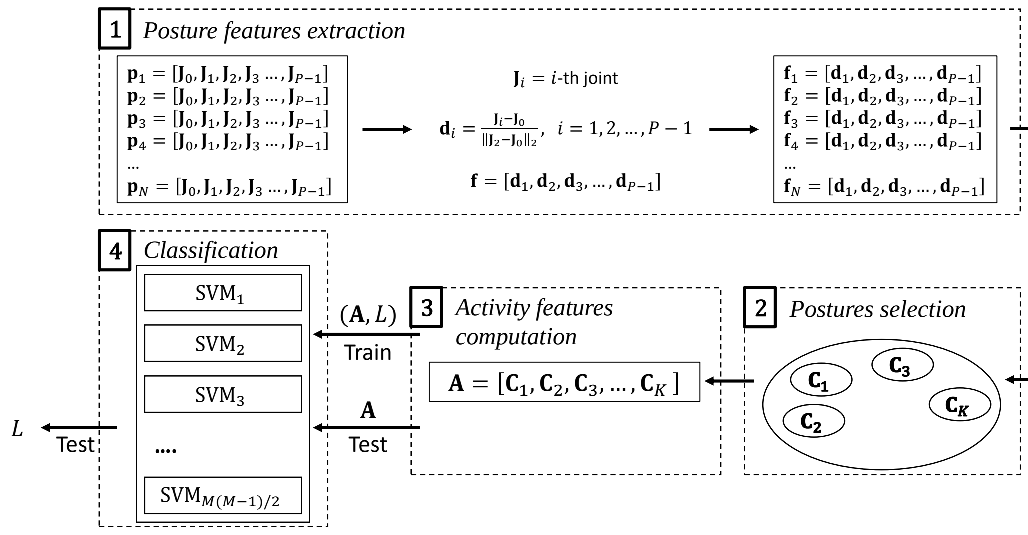

1 Posture features extraction
$\mathbf{p}_1 = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{P-1}]$
$\mathbf{p}_N = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{P-1}]$
$\mathbf{J}_i = i$-th joint
$\mathbf{d}_i = \frac{\mathbf{J}_i - \mathbf{J}_0}{\|\mathbf{J}_2 - \mathbf{J}_0\|_2}, \; i = 1, 2, \dots, P-1$
$\mathbf{f} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}]$
$\mathbf{f}_1 = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}]$
$\mathbf{f}_N = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}]$
2 Postures selection
$\mathbf{C}_1$ $\mathbf{C}_2$ $\mathbf{C}_3$ $\mathbf{C}_K$
3 Activity features computation
$\mathbf{A} = [\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \dots, \mathbf{C}_K]$
4 Classification
SVM_1
SVM_2
SVM_3
....
$\mathrm{SVM}_{M(M-1)/2}$
$(\mathbf{A}, L)$ Train
$\mathbf{A}$ Test
L Test

Figure 5.1.: Main scheme of the HAR algorithm based on Activity Features Vector (AFV).

5.1. HAR algorithm based on activity feature vectors

The HAR algorithm based on activity feature vectors starts from the coordinates of skeleton joints provided by Kinect and initially computes features related to the posture. A clustering algorithm extracts the most informative postures, and a vector with features related to the activity is built. A multi-class SVM is exploited to recognize different activities of the dataset. The AFV algorithm has been tested on five publicly available datasets with a discussion about its performance in activities related to AAL.

5.1.1. Algorithm description

The algorithm based on activity feature vectors comprises four main steps, represented in Figure 5.1. It is organized as most of the machine learning approaches for action recognition, where, first of all, it is necessary to extract features from input data. In the case of RGB-D sensors, the input data can be the streams available from the device but very often they are just skeleton coordinates. Following the feature extraction process, there are usually one or more steps dedicated to the representation of the action. They are aimed to organize features in order to enhance the separation among different classes, i.e. actions. The final phase is always related to classification, and a machine learning algorithm is trained with samples of known actions and then tested with unknown sequences. The four steps of the AFV algorithm are detailed in the following:

1. *Posture features extraction*: feature vectors representing human postures are extracted from skeleton joints;
2. *Postures selection*: for each sequence constituting an activity, the most representative postures are computed;
3. *Activity features computation*: an activity feature vector, which encloses the information of the whole activity, is created and used for classification;
4. *Classification*: a multi-class SVM implemented with the "one-versus-one" approach is the selected classifier.

The computation of posture features from skeleton data does not include the time information, to ensure the independence from the speed of movement. The aim of this step is to obtain features related to posture which are independent from the position of the skeleton within the coverage area of Kinect, and also from the build of the subject. The compensation of the skeleton position is achieved by centering the coordinate space in one skeleton joint while the processing related to build is achieved by normalizing the features with respect to the length of human torso. By defining as $\mathbf{J}_i = [J_i^x, J_i^y, J_i^z]$ the vector representing the coordinates of the i-th skeleton joint at a specific frame of an action and being $\mathbf{J}_0$ the coordinates of the torso joint and $\mathbf{J}_2$ the coordinates of the neck joint, $\mathbf{d}_i$ is the i-th joint feature and it is the position difference between $\mathbf{J}_i$ and $\mathbf{J}_0$, normalized to the ℓ_2-norm between $\mathbf{J}_2$ and $\mathbf{J}_0$:

$$\mathbf{d}_i = \frac{\mathbf{J}_i - \mathbf{J}_0}{\|\mathbf{J}_2 - \mathbf{J}_0\|_2}, \qquad i = 1, 2, ..., P-1 \tag{5.1}$$

assuming that a skeleton is made by P joints. These features may be seen as a set of vectors connecting each skeleton joint to $\mathbf{J}_0$. A posture feature vector $\mathbf{f}$, with $P-1$ vectors of three dimensions each, represents a single skeleton frame:

$$\mathbf{f} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}] \tag{5.2}$$

As shown in Figure 5.1, having a set of N skeleton frames $[\mathbf{p}_1, \mathbf{p}_2, \dots \mathbf{p}_N]$ requires the computation of N feature vectors $[\mathbf{f}_1, \mathbf{f}_2, \dots \mathbf{f}_N]$. A skeleton integrity check is included in the feature extraction process and, if all the skeleton joints are unavailable for the i-th frame, the posture feature vector $\mathbf{f}_{i-1}$ is associated also to $\mathbf{f}_i$.

The second phase aims to select the most important postures among the set of N posture features vectors representing the action. This process allows to reduce the complexity and to increase generality by representing the activity with a reduced subset of poses, without using all the N frames. The N feature vectors constituting the activity can be grouped into K clusters with the k-means clustering algorithm, exploiting the squared euclidean distance as a metric. If an activity is represented by the set $[\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \ldots, \mathbf{f}_N]$, the k-means algorithm partitions data into K clusters $S_1, S_2, \ldots, S_K$ satisfying the condition expressed by (5.3):

$$\arg\min_{\mathbf{S}} \sum_{j=1}^{K} \sum_{\mathbf{f}_i \in S_j} \|\mathbf{f}_i - \mathbf{C}_j\|^2 \tag{5.3}$$

where $[\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \ldots, \mathbf{C}_K]$ are the K centroids of the clusters which can be considered as the most important feature vectors constituting the sequence, i.e. the selected postures. The clustering algorithm which selects the main postures is executed for each sequence of frames, both in training and testing phase, selecting a number of postures which is N at maximum.

The activity features vector modelling the entire activity is computed in the third phase of the algorithm. The centroid vectors $[\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \ldots, \mathbf{C}_K]$, sorted considering the order in which the cluster's elements occur during the activity, constitute the activity features vector. For an activity with $N = 8$ and choosing $K = 3$, the k-means algorithm could assign the vectors $[\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \ldots, \mathbf{f}_8]$ to the following sequence of cluster IDs: $[2, 2, 2, 3, 3, 3, 1, 1]$. In this specific case, the activity features vector would be $\mathbf{A} = [\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_1]$. Multiple repetitions of the same cluster ID are discarded, thus $\mathbf{A}$ has a fixed dimension of $3K(P-1)$.

The aim of the classification phase is to distinguish between activity features vectors belonging to different classes. In the training phase the classifier takes the output from the previous step ($\mathbf{A}$) and a label (L) identifying the class of the dataset. During testing, it should be able to associate each $\mathbf{A}$ to the corresponding L. Considering a binary SVM, trained with a number of l vectors $\mathbf{x}_i \in R^n$ and a vector of labels $y \in R^l$, where $y_i \in \{-1, 1\}$, the following equations describe the classification problem [152]:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i\left(\mathbf{w}^T\phi\left(\mathbf{x}_i\right)+b\right) \geq 1-\xi_i, \\ & \xi_i \geq 0, i = 1,\ldots,l \end{aligned} \tag{5.4}$$

where

$$\mathbf{w}^T\phi\left(\mathbf{x}\right)+b=0 \tag{5.5}$$

is the hyperplane that ensures optimal separation between training vectors in the feature space, C is the parameter to set the dimension of the separation margin, and ξ_i are slack variables which take into account training errors. The function ϕ allows mapping inputs into high-dimensional space where the data are separable. Given two training vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, the kernel function ϕ is given by:

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\phi\left(\mathbf{x}_i\right)^T\phi\left(\mathbf{x}_j\right) \tag{5.6}$$

Several kernel functions can be used, among which the Radial Basis Function (RBF), defined as:

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}, \quad \gamma>0 \tag{5.7}$$

From the definition of SVM with RBF kernel, it derives that C and γ are the parameters that have to be tuned to use the SVM.

Even if SVMs were originally proposed as binary classifiers, allowing to perform separation between 2 classes, different methods have been proposed to apply these algorithms with more classes. The chosen approach is known as “one-versus-one”, which is based on the definition of $M(M-1)/2$ binary SVMs for an algorithm able to recognize M classes. Each SVM is trained to distinguish between 2 classes and the classification is done through a voting strategy, where all the classifiers select one class and the one with more votes is the output class.

5.1.2. Experimental results

The performance of AFV algorithm is evaluated on five publicly available datasets: KARD, CAD-60, UTKinect, Florence3D, MSR Action3D, described in Subsection 2.2.3. The algorithm requires the definition of the number and configuration of joints that have to be considered by the algorithm. Different subsets, going from

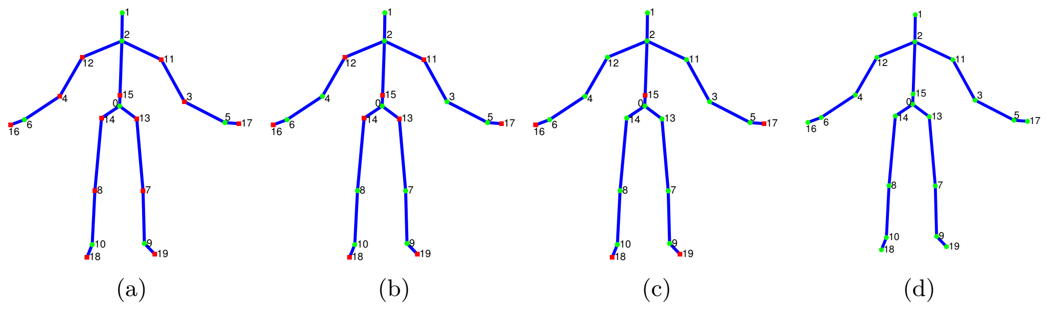


Figure 5.2.: Subsets of 7 (a), 11 (b), 15 (c) joints, and the whole set of 20 joints (d), considered to evaluate the algorithm. Green circles are the selected joints and red squares are the discarded ones.

a minimum of 7 up to the maximum of 20 joints have been evaluated. Figure 5.2 shows the 4 subsets, where the selected joints are depicted as green circles while the discarded ones are red squares. The performance in AAL scenarios are also evaluated by selecting only some subsets of activities which are related to AAL. The algorithm has been implemented in MATLAB, using the multi-class SVM implementation provided by LIBSVM library [153].

KARD dataset

Considering KARD dataset [90], tests are executed with the different splitting modalities described in Subsection 2.2.3, which consider separately Gestures and Actions or the three activity subsets in Table 2.3. Furthermore, the Experiments A, B, C (also described in Subsection 2.2.3) which split data between training and testing are also considered and each of them has been repeated 10 times, randomly splitting training and testing data. Finally, the leave-one-actor-out cross-validation scheme on the whole dataset is also executed. Clusters belonging to the set $K = [3, 5, 10, 15, 20, 25, 30, 35]$ have been considered for each test conducted on KARD dataset. In addition to the number of postures per sequence, the different subsets of skeleton joints represented in Figure 5.2 are considered, with the exception of the whole skeleton ($P = 20$) because only 15 skeleton joints are available.

The accuracy obtained considering the three Activity Sets is shown in Table 5.1; the implemented AFV algorithm outperforms the one designed by Gaglio et al. [90], whom collected dataset, in almost all the tests. The maximum accuracy is achieved with a number of postures (K) which is 30 or 35 in most of the cases. It means that,

Table 5.1.: Accuracy (%) of AFV algorithm compared to other using KARD dataset with different Activity Sets, and for Experiments A, B, C.

	Activity Set 1			Activity Set 2			Activity Set 3		
	A	B	C	A	B	C	A	B	C
Gaglio et al. [90]	95.1	**99.1**	93.0	89.9	94.9	90.1	84.2	89.5	81.7
AFV ($P = 7$)	**98.2**	98.4	**98.1**	99.7	**100**	**99.7**	90.2	95.0	91.3
AFV ($P = 11$)	98.0	99.0	97.7	**99.8**	**100**	99.6	**91.6**	**95.8**	**93.3**
AFV ($P = 15$)	97.5	98.8	97.6	99.5	**100**	99.6	91	95.1	93.2

for the KARD dataset where the number of frames constituting each activity goes from a minimum of 42 in a sequence of *Hand clap* gesture, to a maximum of 310 for a *Walk* sequence, it is better to have a considerable number of clusters to model an activity.

Experiment A, which requires to consider one third of the data for training and the remaining for testing, is influenced a lot by the number of clusters in Activity Set 1. The maximum accuracy (98.2%) shown in Table 5.1 is achieved with $P = 7$ and $K = 35$. By reducing the number of clusters to $K = 5$, the minimum accuracy is 91.4% with a difference of 6.8%. By considering more data for training, the dependence on the number of postures is reduced. Experiment B, which consists in two third of the data for training and one third for testing, is featured by a gap of 0.5%, 1.1% and 2.6% respectively in Activity Set 1, Activity Set 2 and Activity Set 3.

Considering the performance with different sets selected joints, Table 5.1 shows that Activity Set 1 and 2, which are the simplest ones, have good recognition results using the smaller subset with $P = 7$ joints. The entire set of joints is not required neither in Activity Set 3, composed by more similar activities, where slightly better results are obtained with $P = 11$.

Table 5.2 lets to conclude that the AFV method outperforms Gaglio et al. [90] also when the dataset is split in Gestures and Actions. Better results are obtained considering the Actions subset and, even in this case, it is not necessary to consider all the skeleton joints, the best recognition results are given by $P = 7$ and $P = 11$ with $K = 30$ or $K = 35$.

Table 5.3 compares the results obtained with the leave-one-actor-out cross-validation test in terms of precision and recall. In this setting, the entire dataset of 18 activ-

Table 5.2.: Accuracy (%) of AFV algorithm compared to other using KARD dataset, with dataset split in Gestures and Actions, and for experiments A, B, C.

	Gestures			Actions		
	A	B	C	A	B	C
Gaglio et al. [90]	86.5	93.0	86.7	92.5	95.0	90.1
AFV ($P = 7$)	**89.9**	93.5	92.5	**99.1**	99.6	**99.4**
AFV ($P = 11$)	**89.9**	**95.9**	**93.7**	99.0	**99.9**	99.1
AFV ($P = 15$)	87.4	93.6	92.8	98.7	99.5	99.3

Table 5.3.: Precision (%) and Recall (%) of AFV algorithm compared to other, using the whole KARD dataset and leave-one-actor-out cross-validation setting.

Algorithm	Precision	Recall
Gaglio et al. [90]	84.8	84.5
AFV ($P = 7$)	94.0	93.7
AFV ($P = 11$)	**95.1**	**95.0**
AFV ($P = 15$)	95.0	94.8

ities is considered, and the best result is obtained with $K = 30$ and $P = 11$. The AFV algorithm reaches about 95% of precision and recall outperforming the results obtained by Gaglio et al. [90], which are about 85%. The confusion matrix obtained from this test is shown in Figure 5.3. Most of the activities constituting the dataset are recognized with an accuracy higher than 90%. Lower percentages are achieved for the gestures *Phone Call* and *Drink* and also *Draw X* and *Draw Tick* which are quite similar and can be mixed each others.

Considering the application of the AFV algorithm in the AAL domain with an evaluation on KARD dataset, only the activities which are part of the Action subset can be relevant, i.e. *Catch Cap, Toss Paper, Take Umbrella, Walk, Phone Call, Drink, Sit Down, Stand Up*. This subset, as shown in Table 5.2, can be recognized with an accuracy greater than 98%, even if only one third of data is used for training (Experiment A) and considering that there are similar actions such as *Sit down* and *Stand up*, or *Phone Call* and *Drink*.

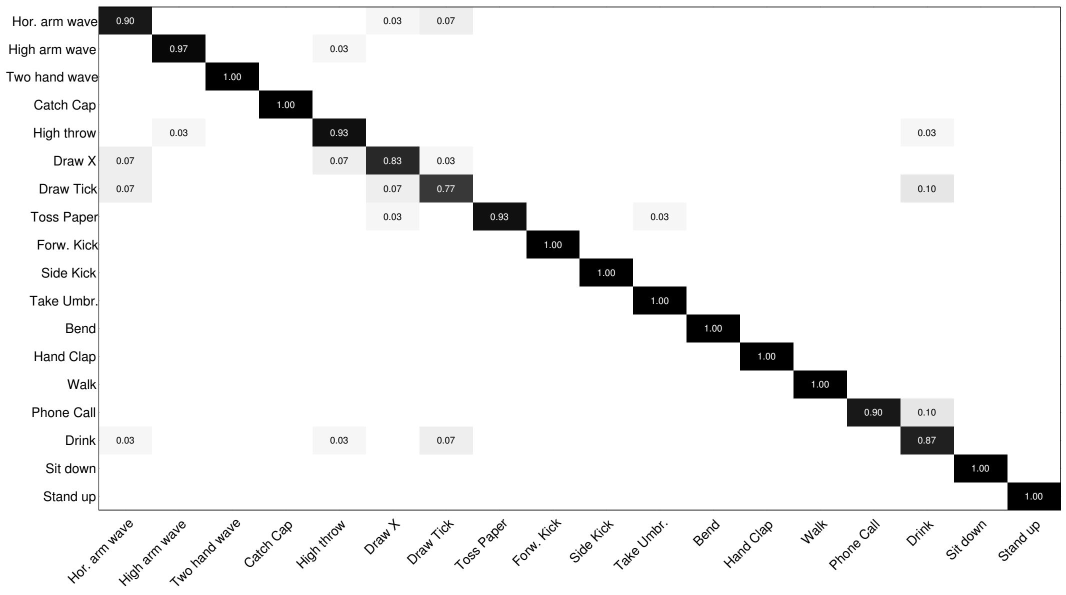


Figure 5.3.: Confusion matrix of the AFV algorithm obtained with leave-one-actor-out cross-validation test on the whole KARD dataset.

CAD-60 dataset

The CAD-60 dataset [98], as described in Subsection 2.2.3, is a dataset constituted by activities performed in 5 different environments by 4 people. The 12 activities are considered separately as a function of the environment and the global performance of the algorithm is given by the average precision and recall over all the environments. The most used setting is the so-called "new-person", which is a leave-one-actor-out cross-validation.

This dataset is challenging because one of the four actor is left-handed, which can be a problem for some activities involving one-hand gestures. In order to make the algorithm independent to the main hand of the subject, mirrored copies of actions performed by left-handed and right-handed actors are created, as suggested in [98]. For each action, a dummy version has been obtained by mirroring the skeleton with respect to the virtual sagittal plane that cuts the person in a half. As in KARD dataset, the number of available joints for CAD-60 is $P = 15$, thus only the first three sets of joints in Figure 5.2 can be considered for experiments, with a number of clusters belonging to the sequence $K = [3, 5, 10, 15, 20, 25, 30, 35]$.

The combination of parameters $P = 11$ and $K = 25$ gives the best results, which are shown in Table 5.4 in terms of precision and recall for each activity. Considering

Table 5.4.: Precision (%) and Recall (%) of AFV algorithm for different environments of CAD-60 and leave-one-actor-out cross-validation setting, with $P = 11$ and $K = 25$.

Location	Activity	Precision	Recall
bathroom	brushing teeth	88.9	100
	rinsing mouth	92.3	100
	wearing contact lens	100	79.2
	Average	**93.7**	**93.1**
bedroom	talking on phone	91.7	91.7
	drinking water	91.3	87.5
	opening pill container	96.0	100
	Average	**93.0**	**93.1**
kitchen	cooking (chopping)	85.7	100
	cooking (stirring)	100	79.1
	drinking water	96.0	100
	opening pill container	100	100
	Average	**95.4**	**94.8**
living room	talking on phone	87.5	87.5
	drinking water	87.5	87.5
	talking on couch	88.9	100
	relaxing on couch	100	87.5
	Average	**91.0**	**90.6**
office	talking on phone	100	87.5
	writing on whiteboard	100	95.8
	drinking water	85.7	100
	working on computer	100	100
	Average	**96.4**	**95.8**
	Global Average	**93.9**	**93.5**

Table 5.5.: Global Precision (%) and Recall (%) of AFV algorithm for CAD-60 dataset and leave-one-actor-out cross-validation setting, with different subsets of joints, compared to other works.

Algorithm	Precision	Recall
Sung et al. [98]	67.9	55.5
Gaglio et al. [90]	77.3	76.7
AFV ($P = 15$)	87.9	86.7
Faria et al. [154]	91.1	91.9
Parisi et al. [155]	91.9	90.2
AFV ($P = 7$)	92.7	91.5
Shan and Akella [156]	93.8	**94.5**
AFV ($P = 11$)	**93.9**	93.5

separately the different environments, the best results have been obtained in *office* environment, where the average precision and recall are 96.4% and 95.8% respectively. The reason is that the activities of this environment are quite different, apart from *talking on phone* and *drinking water* that are similar. On the contrary, the *living room* environment is the most challenging one, since the average precision and recall are 91% and 90.6%. It includes 2 couples of actions that involve quite similar movements: *talking on couch-relaxing on couch* and *talking on phone-drinking water*.

Table 5.5 shows the comparison between the AFV algorithm and other methods validated with the leave-one-actor-out cross-validation setting. AFV algorithm with $P = 11$ configuration outperforms the state of the art results in terms of precision (93.9%), and it is only 1% lower in terms of recall (93.5% against 94.5%). Very good results using a multi-class SVM scheme with a linear kernel have been achieved in [156]. One main difference with the proposed approach is that they train and test mirrored actions separately, and then merge the results when computing average precision and recall. AFV algorithm considers two copies of the same action (the original and the mirrored one) and it gives all the data to the multi-class SVM.

As can be noticed in Table 5.5, changing the parameters of the algorithm may affect the performance in CAD-60 dataset. Using all the available jonts, i.e. $P = 15$, brings to a quite important reduction of the performance, which is 87.9% and 86.7%

for precision and recall. On the other hand, considering a reduced number of joints, as the set shown in Figure 5.2(a) with $P = 7$, does not affect too much the average performance of the algorithm, that achieves a precision of 92.7%, and a recall of 91.5%. Independently from the number of selected joints, better results have been obtained with a high number of clusters ($K = 25$ or $K = 30$). The number of clusters may also affect the results: for the $P = 11$ subset, the adoption of $K = 5$ brings to a precision of 86.6% and a recall of 86.0%, with a gap of 7.3% and 7.5% respectively.

This dataset does not include gestures, and all the actions can be considered of interest from the AAL point of view. Thus, activities of CAD-60 dataset do not have to be split to consider the application in AAL scenario.

UTKinect Dataset

The UTKinect dataset [85], composed by 10 activities executed twice by 10 subjects, is evaluated considering the leave-one-out cross-validation (LOOCV) setting, which consists in training the algorithm with all the sequences except one, and testing it with the the last one. The random effect of the clustering method is mitigated by repeating each training/testing procedure 20 times, and considering the average values as performance indicators, as suggested in [85]. This dataset has been collected using Microsoft SDK and provides 20 skeleton joints, thus all the subsets shown in Figure 5.2 are included in the validation process. The considered sequence of clusters is $K = [3, 4, 5]$, because the sequences have a minimum length of 5 and a maximum length of 120 frames.

The obtained results are shown in Table 5.6, where also previously published works are included. AFV algorithm performs better with the simplest set of joints ($P = 7$), reaching the accuracy of 95.1% when $K = 4$. This result is only 2% lower than the state-of-the-art despite the system is limited by the low number of frames constituting the shortest sequence. Since the considered number of clusters are quite similar, the variation of accuracy changing the parameter K is very low, in particular it is 0.6% with $K = 5$, that generates the worst result. Similar performance has been achieved considering different sets of joints. The accuracy changes only by a 2% going from $P = 15$ (93.1%) to $P = 7$ (95.1%).

The evaluation of UTKinect dataset in AAL scenario requires the consideration of a subset of activities, which is constituted only by actions, without gestures. Only 5 activities out of 10 have been selected: *walk, sit down, stand up, pick up, carry.* Considering again LOOCV setting the performance are slightly better than

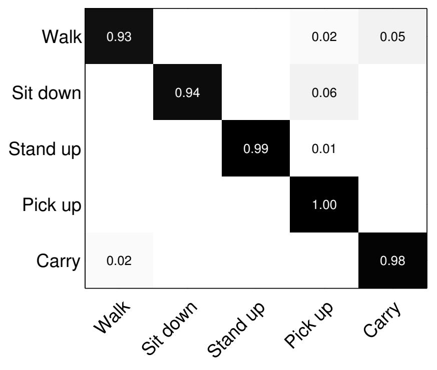
Table 5.6.: Global Accuracy (%) of AFV algorithm for UTKinect dataset and LOOCV setting, with different subsets of joints, compared to other works.

Algorithm	Accuracy
Xia et al. [85]	90.9
Theodorakopoulos et al. [91]	90.95
Ding et al. [92]	91.5
Zhu et al. [74]	91.9
Jiang et al. [114]	91.9
Gan and Chen [83]	92.0
Liu et al. [81]	92.0
AFV ($P = 15$)	93.1
AFV ($P = 11$)	94.2
AFV ($P = 19$)	94.3
Anirudh et al. [95]	94.9
AFV ($P = 7$)	95.1
Vemulapalli et al. [94]	**97.1**

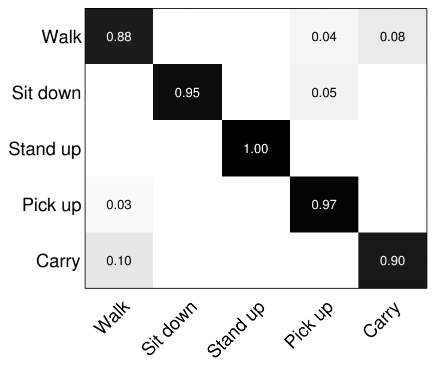
considering the entire dataset. The best results can be achieved with $P = 7$ and $K = 3$ (96.7% accuracy) while worst results are given by $P = 15$ and $K = 3$ (94.1%). Figure 5.4 shows the confusion matrices obtained from the best and the worst configurations. The consideration of a lower number of joints, apparently, brings to a better discrimination between the activities *walk* and *carry*. The inclusion of more joints ($P = 15$) introduces noise leading to a higher misclassification between the two similar classes.

Florence3D Dataset

The Florence3D dataset [105] provides data of 9 activities performed by 10 people multiple times, resulting in a total number of 215 sequences. The results on this dataset are compared with the leave-one-actor-out method considering three subsets of skeleton joints, from $P = 7$ to $P = 15$. The sequence of clusters $K = [3, 4, 5, 6, 7, 8]$ has been considered to build the activity feature vector, considering that the mini-



(a)



(b)

Figure 5.4.: Confusion matrices of the UTKinect dataset with only AAL related activities. (a) Best results ($P = 7$). (b) Worst results ($P = 15$).

mum sequence length is 8.

The results obtained with different subsets of joints are shown in Table 5.7, where the best result for the AFV algorithm is 86.1% given by $P = 11$, with 6 clusters. As in UTKinect, the number of clusters are quite similar and the choice of K affects the performance only for a percentage of 2%. State-of-the-art accuracy is achieved by Taha et al. [89] and it is 96.2%.

A gap of 4% is present considering different number of joints, with the worst result of 82.1% represented by $P = 7$ and $K = 3$. The inclusion of all the available joints ($P = 15$) allows to achieve an accuracy which varies between 84.0% and 84.7% by changing the number of postures.

The main challenges raised by Florence3D dataset are high inter-class similarity and intra-class variability. Some actions, for example *drink from a bottle, answer phone, read watch*, are very similar to each other. Figure 5.5 shows some frames constituting the whole sequences and all of them consist in an uprising arm movement which goes to the mouth, hear or head. Intra-class variability is given by different subjects performing the same action in different ways. Some actors use for example left or right hand indifferently to perform *drink from a bottle* or *answer phone* actions.

Performing the analysis on AAL scenario requires the selection of 6 activities out of 9: *drink, answer phone, tight lace, sit down, stand up*. An improvement of the accuracy (90.8%) is achieved with $P = 11$ and $K = 3$, giving the confusion matrix shown in Figure 5.6(a). The worst result is represented by the confusion matrix in

Table 5.7.: Global Accuracy (%) of AFV algorithm for Florence3D dataset and leave-one-actor-out cross-validation setting, with different subsets of joints, compared to other works.

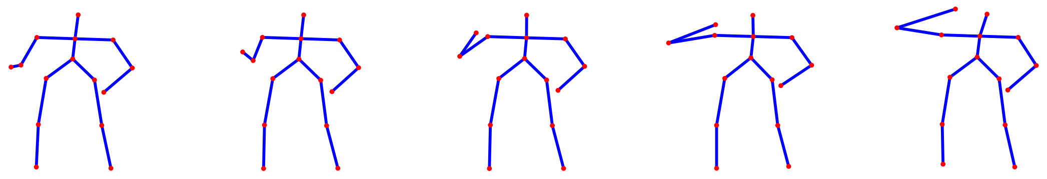
Algorithm	Accuracy
Seidenari et al. [105]	82.0
AFV ($P = 7$)	82.1
AFV ($P = 15$)	84.7
AFV ($P = 11$)	86.1
Anirudh et al. [95]	89.7
Vemulapalli et al. [94]	90.9
Taha et al. [89]	**96.2**

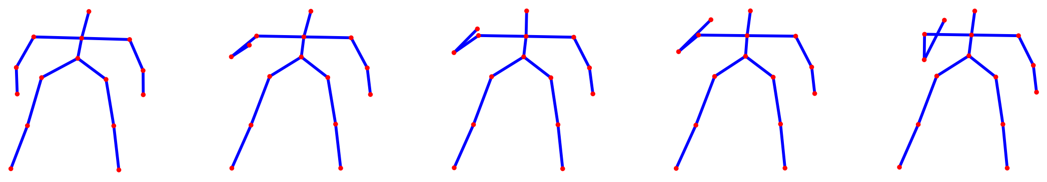
Figure 5.6(b), where the average accuracy of 79.8% is given by $P = 7$ and $K = 4$. The main challenge of this subset is given by the similarity of *drink* and *answer phone* activities.

MSR Action3D Dataset

The MSR Action3D dataset [68] gives 20 actions but it is usually evaluated by considering the subsets AS1, AS2 and AS3 described in Subsection 2.2.3. The evaluation method considered for AFV algorithm is leave-one-actor-out cross-validation. The shortest sequence is constituted by 13 frames, thus the following set of clusters has been considered: $K = [3, 5, 8, 10, 13]$. Since a complete skeleton is provided, all the configurations shown in Figure 5.2, from $P = 7$ to $P = 20$, are considered in the tests.

The proposed AFV algorithm has been tested separately on the three subsets of MSR Action3D, reaching an average accuracy of 81.2% with $P = 7$ and $K = 10$. Figure 5.7 shows the confusion matrices for the three subsets, highlighting that the most difficult subset is AS2 (Figure 5.7(b)), mainly represented by drawing gestures, such as *Draw x* (a07), *Drax tick* (a08) and *Draw circle* (a09). AS1 is also complex because many gestures and actions are quite similar: *Hammer* (a03) and *Forward punch* (a05) for example. Better results are obtained in AS3 (Figure 5.7(c)), where the most difficult actions to recognize are: *Golf swing* (a19) and *Pickup & throw* (a20). Including more joints in the algorithm evaluation process leads to slightly


(a)


(b)

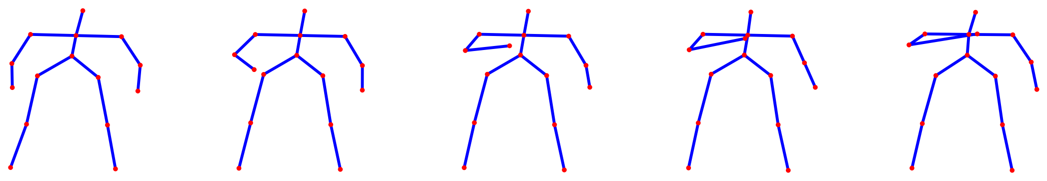

(c)

Figure 5.5.: Sequences of frames representing similar activities from Florence3D dataset. (a) *drink from a bottle*, (b) *answer phone* and (c) *read watch*.

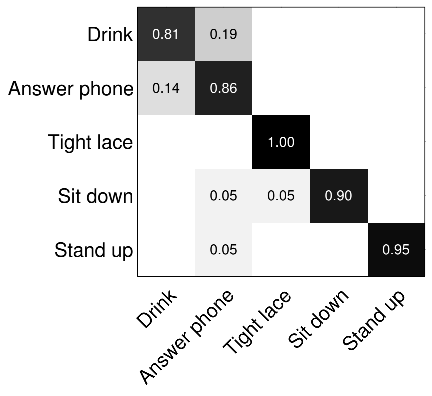
worse results, as shown in Table 5.8. Mid state-of-the-art results are achieved if the AFV algorithm is compared to other approaches available in literature, tested on the leave-one-actor-out cross-validation setting, as shown in Table 5.9.

Since MSR Action3D is mainly constituted by gestures, the evaluation on AAL scenario is not considered for this dataset.

5.2. HAR algorithm based on temporal pyramid of key poses

The algorithm based on activity features vector presented in the previous section has to be reviewed to improve the performance where the dataset is made by actions with similar gestures, such as MSR Action3D.

The HAR algorithm based on temporal pyramid of key poses considers skeleton joints and extracts features representing the person's posture. The adoption of the bag of key poses model [78] allows to learn the most informative postures by creating a codebook. A temporal pyramid is considered to model the temporal structure



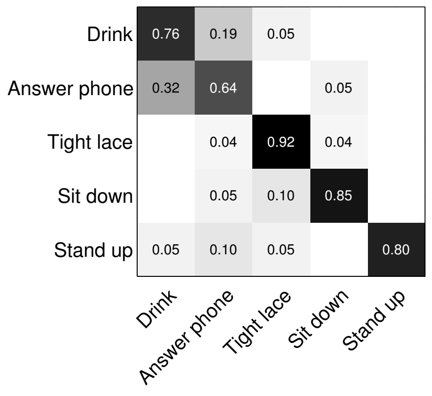


Figure 5.6.: Confusion matrices of the Florence3D dataset with only AAL related activities. (a) Best results ($P = 11$). (b) Worst results ($P = 7$).

Table 5.8.: Accuracy (%) of AFV algorithm for MSR Action3D dataset and leave-one-actor-out setting, with different subsets of joints and activities.

Algorithm	AS1	AS2	AS3	Avg.
AFV ($P = 15$)	78.5	68.8	92.8	80.0
AFV ($P = 20$)	79.0	70.2	91.9	80.4
AFV ($P = 11$)	77.6	73.7	91.4	80.9
AFV ($P = 7$)	79.5	71.9	92.3	**81.2**

of the sequence of frames constituting an action, which is then represented as histograms of key poses. A multi-class SVM is finally exploited to classify actions. The parameters required to run the HAR algorithm are optimized using evolutionary and cooperative coevolutionary algorithms proposed in [115] and [117]. Experimental results have been obtained for MSR Action3D and NTU RGB+D datasets.

5.2.1. Algorithm description

Similarly to the algorithm exploiting activity features vector, 4 main steps constitute the method based on temporal pyramid of key poses. It also adopts the usual approach for action recognition but, if compared to the AFV algorithm, the main differences are in the steps 2 and 3, where the skeleton-based features are organized.

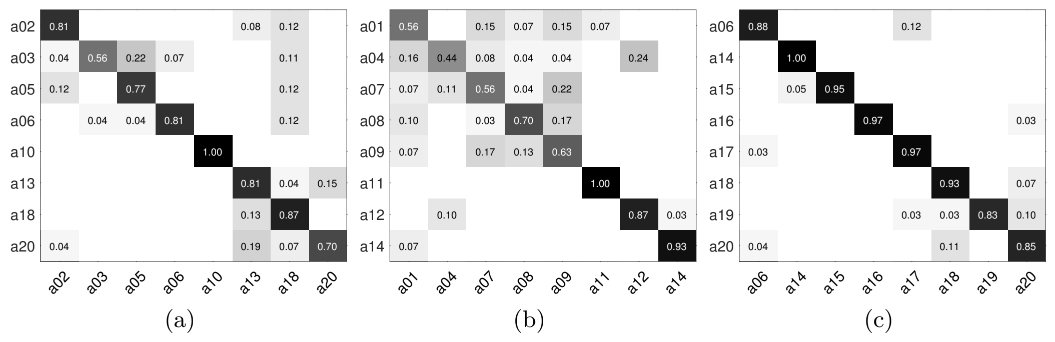


Figure 5.7.: Confusion matrices obtained for AS1 (a), AS2 (b) and AS3 (c) of MSR Action3D dataset, with $P = 7$ and $K = 10$.

Table 5.9.: Average accuracy (%) of AFV algorithm for MSR Action3D dataset and leave-one-actor-out setting, compared to other works.

Algorithm	Accuracy
Çeliktutan et al. [157]	72.9
Azary and Savakis (2013) [158]	78.5
AFV ($P = 7$)	81.2
Azary and Savakis (2012) [159]	83.9
Chaaraoui et al. (2013) [78]	90.6
Chaaraoui et al. (2014) [115]	**93.5**

As can be noticed from Figure 5.8, the main steps are:

1. *Extraction of posture features*: the 3D coordinates of the joints are the input data from which features representing each posture are computed;

2. *Bag of key poses*: this phase is the bag of key poses model, which generates the codebook from training data and associates the closest key pose to each posture in the sequence;

3. *Histograms of key poses and temporal pyramid*: a set of histograms are calculated for each level of a temporal pyramid and they are used to represent a sequence of key poses;

4. *Classification*: a multi-class SVM obtained with "one-versus-one" approach

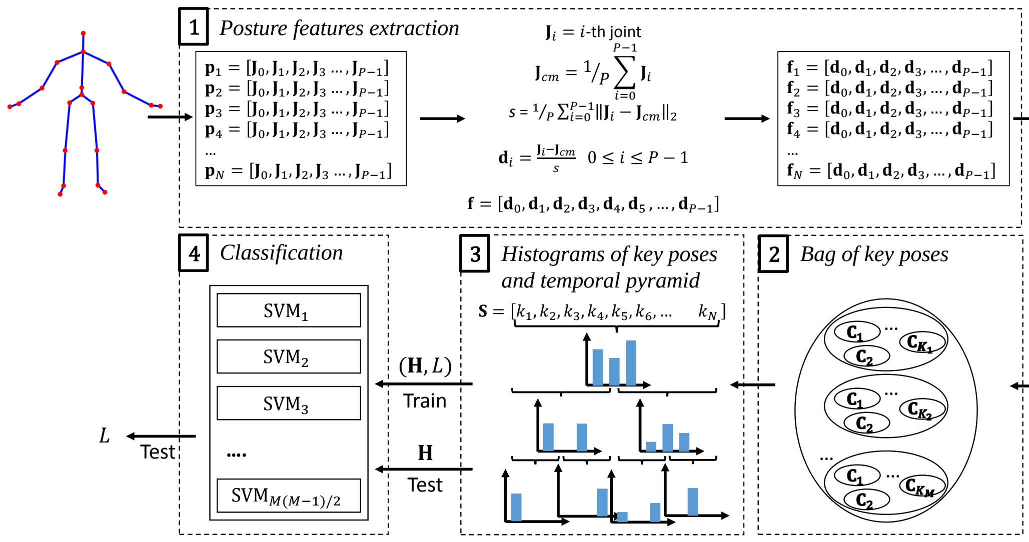

1 Posture features extraction
$\mathbf{p}_1 = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{P-1}]$
$\mathbf{p}_N = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{P-1}]$
$\mathbf{J}_i$ = i-th joint
$\mathbf{J}_{cm} = 1/P \sum_{i=0}^{P-1} \mathbf{J}_i$
$s = 1/P \sum_{i=0}^{P-1} \|\mathbf{J}_i - \mathbf{J}_{cm}\|_2$
$\mathbf{d}_i = \frac{\mathbf{J}_i - \mathbf{J}_{cm}}{s} \quad 0 \le i \le P-1$
$\mathbf{f} = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \dots, \mathbf{d}_{P-1}]$
$\mathbf{f}_1 = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}]$
$\mathbf{f}_N = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{P-1}]$
2 Bag of key poses
3 Histograms of key poses and temporal pyramid
$\mathbf{S} = [k_1, k_2, k_3, k_4, k_5, k_6, \dots \; k_N]$
4 Classification
SVM$_1$
SVM$_2$
SVM$_3$
SVM$_{M(M-1)/2}$
$(\mathbf{H}, L)$
Train
$\mathbf{H}$
Test
L
Test

Figure 5.8.: Main scheme of the HAR algorithm based on temporal pyramid of key poses (TPKP).

takes the histograms of key poses and performs classification.

The first step, as in AFV algorithm, is the extraction of features representing the posture. However, in this version, the reference joint is not the joint of torso, but the center-of-mass $\mathbf{J}_{cm}$ computed by averaging the 3D position of all the P joints:

$$\mathbf{J}_{cm} = \frac{1}{P} \sum_{i=0}^{P-1} \mathbf{J}_i \tag{5.8}$$

The normalization factor, previously represented by the distance between neck and torso, is now computed considering the average ℓ_2-norm between each joint and the center-of-mass:

$$s = \frac{1}{P} \sum_{i=0}^{P-1} \|\mathbf{J}_i - \mathbf{J}_{cm}\|_2 \tag{5.9}$$

and the position difference $\mathbf{d}_i$, related to the i-th joint within the skeleton of P elements, is implemented by updating equation (5.1) with the introduction of $\mathbf{J}_{cm}$:

$$\mathbf{d}_i = \frac{\mathbf{J}_i - \mathbf{J}_{cm}}{s}, \qquad i = 0, 1, 2, ..., P-1 \tag{5.10}$$

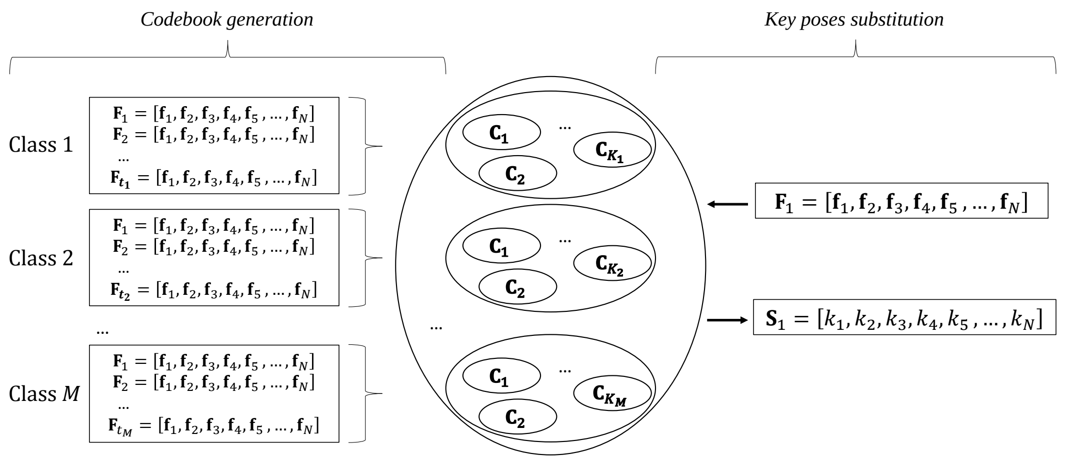

Codebook generation
Key poses substitution
Class 1
Class 2
Class M
$\mathbf{F}_1 = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \dots, \mathbf{f}_N]$
$\mathbf{F}_2 = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \dots, \mathbf{f}_N]$
$\mathbf{F}_{t_1} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \dots, \mathbf{f}_N]$
$\mathbf{F}_{t_2} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \dots, \mathbf{f}_N]$
$\mathbf{F}_{t_M} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4, \mathbf{f}_5, \dots, \mathbf{f}_N]$
$\mathbf{C}_1$
$\mathbf{C}_2$
$\mathbf{C}_{K_1}$
$\mathbf{C}_{K_2}$
$\mathbf{C}_{K_M}$
$\mathbf{S}_1 = [k_1, k_2, k_3, k_4, k_5, \dots, k_N]$

Figure 5.9.: Description of bag of key poses model.

The computation of posture features for the n-th frame in a sequence of N skeletons constituting an action consists in a set of P differences:

$$\mathbf{f}_n = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{P-1}] \tag{5.11}$$

The same skeleton integrity check implemented for AFV algorithm, which associates the feature vector $\mathbf{f}_{n-1}$ to $\mathbf{f}_n$ if the n-th skeleton is not available, is considered also here.

The second step implements the bag of key poses model [115]. It consists of the codebook generation, with the definition of the key poses, and the substitution of each posture features with the most informative feature vectors, i.e. the key poses. This process is represented in Figure 5.9 and the codebook generation phase is realized with a separated clustering process for each action of the dataset using k-means algorithm. A separated clustering is performed because different actions may be better represented by a different number of key poses. Considering a dataset with M actions, i.e. M classes, a vector with M elements $[K_1, K_2, \dots, K_M]$ can be defined with the number of key poses. As visible in Figure 5.9, all the t_1 training instances of the first class $[\mathbf{F}_1, \mathbf{F}_2, \dots, \mathbf{F}_{t_1}]$ are grouped in K_1 clusters, and the cluster centers $[\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_{K_1}]$ are the key poses for action 1. By repeating this process for all the classes and by merging all the key poses obtained for each class, the codebook is obtained. The closest key pose in terms of euclidean distance has to be associated to each feature vector in an action. Considering Figure 5.9, an action represented by a sequence of feature vectors $\mathbf{F}_1 = [\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_N]$ is translated into a sequence of key poses $\mathbf{S}_1 = [k_1, k_2, \ldots, k_N]$. The codebook is generated considering only training data while feature vectors extracted from testing samples have to be associated to learned key poses.

The third step consists in the representation of the action with a temporal pyramid, and the computation of the histograms of key poses at each level of the pyramid. As visually represented in Figure 5.8, a sequence is divided into a number of segments of 2^{l-1} at the l-th level of the pyramid. A histogram is built for each segment of the pyramid considering the number of occurrences of each key pose within the segment, normalized by the sequence length. The temporal pyramid is an efficient structure to represent the distribution of the key poses within the sequence. In fact, moving from the top to the bottom of the pyramid, different descriptions of the same sequence can be obtained, from the most general to the most detailed one. Furthermore, the temporal pyramid can be efficiently implemented because the computation of the histograms at the l-th level of the pyramid can be efficiently obtained considering the sum of the corresponding segments at the level $l+1$. The sequence is finally represented as the histograms computed at each level of the pyramid. Considering the example in Figure 5.8, where a temporal pyramid of 3 levels is shown, the vector $\mathbf{H}$, which is the input to the classifier, is represented by 7 histograms.

Finally, the last step realizes classification and is required to associate each set of histograms $\mathbf{H}$ representing an action to the corresponding class label L. The classifier is exactly the same as in the AFV algorithm, i.e. a multi-class SVM implemented with "one-versus-one" method and RBF kernel.

5.2.2. Parameter optimization

In order to improve the performance in terms of classification accuracy, the parameters required by the algorithm based on temporal pyramid of key poses can be optimized by using evolutionary and coevolutionary algorithms, by adopting the approaches proposed by Chaaraoui et al. [115, 117] to optimize their HAR method. The idea is to find the best values for three parameters of the HAR algorithm: the set of joints to be included in the HAR algorithm (*features*), the number of key poses for each class of the dataset (*clusters*), and the set of training sequences (*instances*).

Evolutionary optimization

Evolutionary algorithms are optimization approaches inspired by biological evolution where the population of individuals (i.e. the candidate solutions) evolve towards

features				*clusters*				*instances*			
1	2	...	*P*	1	2	...	*M*	1	2	...	*I*

Figure 5.10.: Individual defined for evolutionary optimization.

better solution every iteration. An individual is usually defined as a binary array and each gene can be 1 or 0, requiring or not its usage in the HAR algorithm. Since three different parameters have to be optimized, the adopted method considers an individual as a binary array constituted by three parts, each of them related to a different parameter [160]. The optimization algorithm is applied as a wrapper method, by iteratively trying the candidate solutions on the HAR algorithm and evolving the population according to their fitness, i.e. the accuracy of HAR algorithm. The evolutionary computation is constituted by the following steps:

1. the population is initialized with a fixed number of individuals (N);

2. the accuracy of the HAR algorithm is associated to the fitness, and the population is ranked considering the fitness value of each individual;

3. a new individual is generated by applying the crossover operator, considering two parents selected with ranking method.
   This selection method considers the absolute fitness value of each element to order the population, then it assigns a fitness value according to the position of each element. If the population is constituted by N individuals, the best one receives a fitness of N, the second one takes a fitness value of $N - 1$, and so on, until the worst individual which takes the value 1.

4. mutation operator is applied to the new individual;

5. the fitness for the new individual is evaluated and it is added to the population;

6. the next generation population is selected with elitism. All the individuals of the population are selected with the exception of the individual featured by the lowest fitness;

7. the algorithm creates a new individual, repeating the process from step 3 until the stop condition, which is a number of generations without changes in the highest fitness.

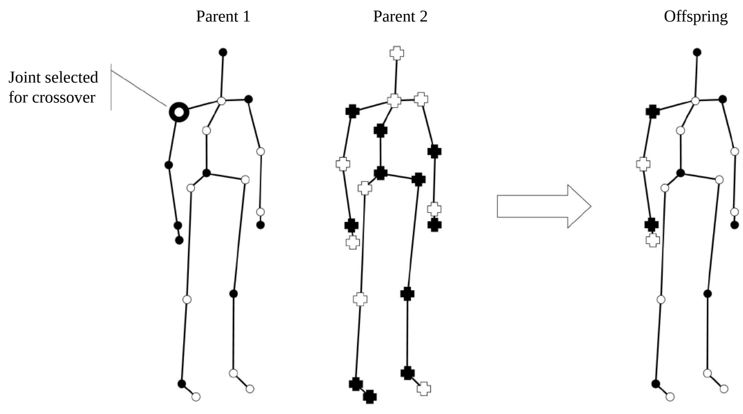


Figure 5.11.: Crossover operator aware of skeleton structure, applied to the *features* sub-individual (reprinted from [115]).

The individual's structure is represented in Figure 5.10, where the *features* item is represented by a binary vector of length P, the *clusters* item is constituted by M integer values (one for each class), and the *instances* sub-individual is made up of I elements, assuming the presence of I training sequences in the dataset. A 1-point crossover operator is applied independently to each part and, while a standard crossover is applied to *instances* and *clusters* vectors, a specific one, which takes into account the skeleton structure, is considered for the *features* part. Given the joint selected for crossover, the specific crossover operator applied to the *features* sub-individual considers the upper joints from one parent and the bottom joints from the other one. As visible in Figure 5.11, if a shoulder joint is selected for crossover, the offspring is constituted by most of the joints from parent 1 with the arm from parent 2. The colors of the joints are associated to the value of the corresponding gene in the individual: black joints assume value 1 and they are selected in the HAR algorithm while white joints assume value 0 and they are discarded.

The mutation operator, which is applied to alter the gene values in the individual from its initial state, is also featured by three probabilities, one for each sub-individual. Probabilities mut_P, mut_M and mut_I are defined for *features* vector, *clusters* vector and *instances* vector. If the sub-individual takes binary values, a gene can change its value according to the corresponding mutation probability. If the sub-individual assumes integer values, for example in the *clusters* vector, mutation is performed by changing the actual value with a random value contained in a specific interval.

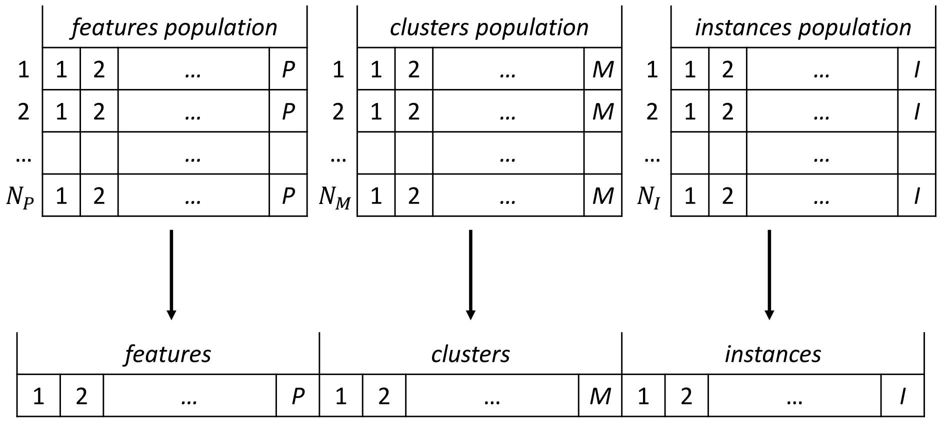


Figure 5.12.: The individual for coevolutionary optimization is obtained by considering three sub-individuals belonging to different populations: *features*, *clusters* and *instances* (reprinted from [117]).

Coevolutionary optimization

In the cooperative coevolutionary algorithm, each sub-individual defined to optimize the HAR algorithm is associated to a different population, bringing to the consideration of the *instances* population, the *clusters* population and the *features* one [117]. The individuals belonging to the three populations have the same structure of the corresponding sub-individuals defined in the case of a unique population in the evolutionary optimization process. Crossover and mutation operators can be applied with the same choices. One difference is represented by the selection of parents required to create a new individual. In this case, the selection is implemented with roulette method, which means that the possibility for each element to be selected is proportional to its fitness.

As represented in Figure 5.12, the fitness value for a new individual of one population (i_1) requires the selection, which is implemented with ranking method, of individuals from the two other populations (i_2, i_3). The obtained fitness value is associated not only to the individual i_1, but also to i_2 and i_3 if it improves the fitness value associated to them. If two or more individuals have the same fitness values, different priorities are given to each of them. Individuals with a lower number of selected values are preferred in *features* and *instances* populations. In the *clusters* population, the individual with less accumulated sum is favored. This mechanism gives priority to solutions that limit the complexity of the system.

Table 5.10.: Accuracy (%) and parameter values obtained considering Random selection, Evolutionary and Coevolutionary optimizations by the TPKP algorithm with MSR Action3D.

	AS1	AS2	AS3
		Random selection	
Acc.	95.24	86.61	95.5
clust.	[17 17 15 25 8 22 12 22]	[4 8 10 22 18 19 16 5]	[71 66 48 56 66 61 76 52]
		Evolutionary optimization	
Acc.	95.24	90.18	100
clust.	[10 26 12 10 17 22 10 10]	[7 13 10 5 9 16 23 17]	[68 69 60 62 55 48 75 60]
feat.	[11100001011110001000]	[11100111110110011111]	[10100101110010100011]
		Coevolutionary optimization	
Acc.	95.24	91.96	98.2
clust.	[15 7 9 12 12 13 5 10]	[10 10 10 5 13 4 10 16]	[51 15 16 34 29 56 55 43]
feat.	[10101001100010001100]	[00001001101110011110]	[11111001011110100011]
inst.	178/219	202/228	176/222

5.2.3. Experimental results

Experimental tests on algorithm based on temporal pyramid of key poses have been executed on the well known MSR Action3D dataset [68], and also on a recently released large-scale dataset: NTU RGB+D [97]. Due to the complexity of the latter dataset, the optimization algorithm based on evolutionary and coevolutionary optimization has been applied only on MSR Action3D.

MSR Action3D

The evaluation scheme adopted for MSR Action3D is the most used one, i.e. the cross-subject test defined by Li et al. [68], which considers samples from actors 1-3-5-7-9 for training, and sequences performed by actors 2-4-6-8-10 for testing. The evaluation is performed considering the accuracy obtained separately for the subsets AS1, AS2, AS3 (Table 2.2) and the average as global performance indicator.

In order to evaluate the effect of evolutionary algorithms, the selection of parameters has been performed using three different methods, all of them considering three levels of the temporal pyramid, detailed in the following:

- *Random selection*: the entire set of training sequences (*instances*) and skeleton joints (*features*) are considered, the only selected parameter is the the number of key poses for each class of the dataset. The *clusters* are selected randomly in the interval $[4, 26]$ when the subsets AS1 and AS2 are evaluated and within the interval $[44, 76]$ for AS3;

- *Evolutionary optimization*: the evolutionary algorithm is applied to select the *features* and the *clusters* sub-individuals, considering the same selection intervals applied in the *random selection*: $[4, 26]$ for AS1 and AS2 and $[44, 76]$ for AS3. The sub-individual regarding the training *instances* is not considered, and all the sequences required by the evaluation scheme are included in the training process. A number of 10 individuals constitute the population and the mutation probabilities are not fixed, but they are randomly selected within the intervals $[0, 0.15]$ for mut_P and $[0, 0.25]$ for mut_C. The optimization process is terminated with 100 generation without changing the best fitness value.

- *Coevolutionary optimization*: the algorithm is adopted to optimize *instances*, *features* and *clusters* sub-individuals. A random selection within the interval $[0, 0.025]$ gives the mutation probability of *instances* vector (mut_I). The *clusters* are randomly chosen within the interval $[4, 16]$ for AS1 and AS2, and $[4, 64]$ for AS3. The termination condition is represented by a fixed number of generations, independently from the variations in the highest fitness. A value of 500 has been used for AS2 and AS3 while AS1 required 800 generations.

Table 5.10 shows the results obtained with the *random selection* of parameters, *evolutionary* and *coevolutionary optimization*. Considering the *random selection* of clusters representing key poses, the algorithm can overcome 95% accuracy in AS1 and AS3, while it reaches 86.61% in AS2 which is the most complex subset, as revealed also by other methods. However, probably due to the inclusion of more complex actions, AS3 subset requires an higher number of key poses per class, if compared to AS1 and AS2. The optimization with the evolutionary algorithm, considering the number of clusters per class, and the set of skeleton joints, brings to an improvement of the performance in AS2 and AS3. The latter can be recognized with 100% score even if it requires a large number of key poses, which can be 75 at maximum for the *Golf swing* action. On the other hand, AS3 requires a quite limited number of joints: only 10 joints out of 20 are necessary in the best configuration. The best recognition accuracy for AS2 is 90.18%, which can be achieved with a selection of 15 joints and a quite low number of clusters per class, which is maximum 23. The

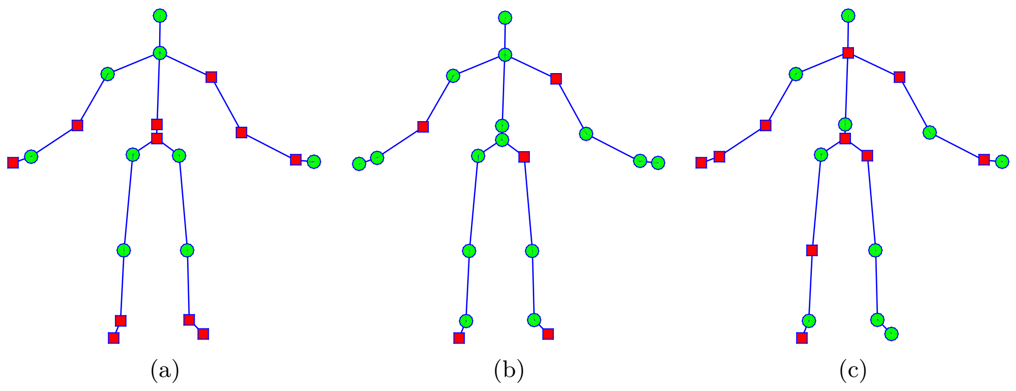


Figure 5.13.: Best subsets of joints selected with evolutionary optimization for AS1 (a), AS2 (b) and AS3 (c). Green circles represent the selected joints while red squares are used for the discarded ones.

highest accuracy obtained for AS1 is the same as in the *random selection* process, but now only 9 joints are required, limiting the complexity of the system. The selected subsets of joints by the *evolutionary optimization* is shown in Figure 5.13. They are consistent with the movements required by the actions included in the subsets: differently from AS2 and AS3, foot and ankle joints are not considered in AS1 because actions do not require movements of lower limbs.

The *coevolutionary optimization* leads to comparable average results to the evolutionary one. Considering AS1, the recognition accuracy is exactly the same, but the system has a lower complexity because it requires only a number of 178 training instances instead of the entire set of 219, 8 skeleton joints and a total number of 83 key poses for the 8 classes of the subset. With *evolutionary optimization*, 117 key poses are required for the best configuration shown in Table 5.10. Better results have been achieved in AS2, represented by a recognition accuracy of 91.96% obtained with a small set of joints (10) and 202 training instances. Finally, AS3 reached the best accuracy of 98.2%, even if the result could be higher by implementing a different termination condition, for example considering more iterations.

The performance obtained by the TPKP algorithm are compared to other HAR algorithms in Table 5.11. This table has been obtained considering only papers that clearly state the use of the cross-subject test with samples from actors 1-3-5-7-9 for training, and the rest for testing. The TPKP method achieves results comparable to the state-of-the-art considering the average accuracy on AS1, AS2 and AS3 subsets. Shahroudy et al. [170], and Xu et al. [169] perform better but they propose more

Table 5.11.: Accuracy (%) obtained by the TPKP algorithm, compared with other works evaluated on MSR Action3D with the cross-subject test.

Method	AS1	AS2	AS3	avg
Li et al. [68]	72.9	71.9	79.2	74.67
Akkaladevi et al. [161]	84	62	80	75.3
Ghorbel et al. [162]	83.08	79.46	93.69	85.41
Evangelidis et al. [163]	88.39	86.61	94.59	89.86
Chen et al. [164]	96.2	83.2	92	90.47
Chaaraoui et al. [78]	92.38	86.61	96.4	91.8
Lo Presti et al. [165]	90.29	**95.15**	93.29	92.91
Tao and Vidal [166]	89.81	93.57	97.03	93.5
Du et al. [167]	93.3	94.64	95.5	94.49
Chen et al. [168]	98.1	92	94.6	94.9
TPKP	95.24	90.18	**100**	95.14
Xu et al. [169]	**99.1**	92.9	96.4	96.1
Shahroudy et al. [170]	–	–	–	**98.2**

complex methods, including also depth data.

NTU RGB+D

In order to evaluate the proposed algorithm on temporal pyramid of key poses (TPKP) on the large-scale NTU RGB+D dataset, some changes are required. Firstly, NTU RGB+D includes both actions and interactions, so some activities performed by two subjects can be part of the dataset. Secondly, being a quite big dataset, the adoption of the "one-versus-one" multi-class SVM implemented in LIBSVM is not efficient.

The first issue has been solved by modifying the posture features vector defined in (5.11) as follows:

$$\mathbf{f}_n = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{BP-1}] \tag{5.12}$$

which consists in evaluating all the displacements for P skeleton joints and B bodies. It may happen that the sequence contains only one skeleton; in this case, the entire

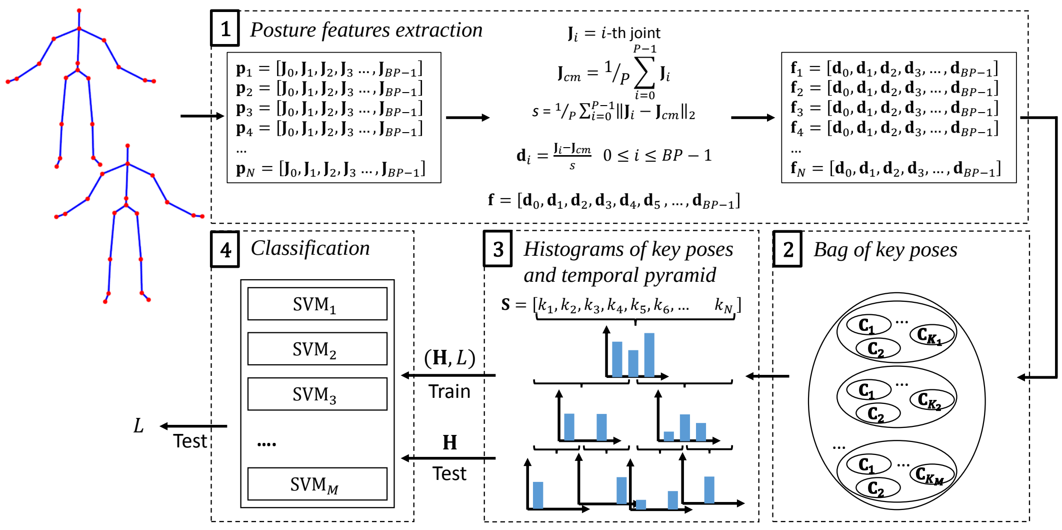

1 Posture features extraction
$\mathbf{J}_i$ = i-th joint
$\mathbf{J}_{cm} = 1/P \sum_{i=0}^{P-1} \mathbf{J}_i$
$s = 1/P \sum_{i=0}^{P-1} \|\mathbf{J}_i - \mathbf{J}_{cm}\|_2$
$\mathbf{d}_i = \frac{\mathbf{J}_i - \mathbf{J}_{cm}}{s} \quad 0 \le i \le BP-1$
$\mathbf{f} = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \dots, \mathbf{d}_{BP-1}]$
$\mathbf{p}_1 = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{BP-1}]$
$\mathbf{p}_N = [\mathbf{J}_0, \mathbf{J}_1, \mathbf{J}_2, \mathbf{J}_3 \dots, \mathbf{J}_{BP-1}]$
$\mathbf{f}_1 = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{BP-1}]$
$\mathbf{f}_N = [\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \dots, \mathbf{d}_{BP-1}]$
2 Bag of key poses
3 Histograms of key poses and temporal pyramid
$\mathbf{S} = [k_1, k_2, k_3, k_4, k_5, k_6, \dots k_N]$
4 Classification
SVM$_1$
SVM$_2$
SVM$_3$
SVM$_M$
$(\mathbf{H}, L)$
Train
$\mathbf{H}$
Test
L
Test

Figure 5.14.: Main scheme of the HAR algorithm based on temporal pyramid of key poses (TPKP) modified for large-scale NTU RGB+D dataset.

features vector is kept to ensure the same dimensionality, but the second half of it is set to zero. On the other hand, the center-of-mass $\mathbf{J}_{cm}$ has been evaluated considering only the P skeleton joints belonging to the main skeleton.

Solving the second issue requires to consider a more efficient approach for large-scale datasets which is the "one-versus-all" multi-class SVM implemented by LIBLINEAR [171] library. At the end, the scheme of TPKP algorithm shown in Figure 5.8 has been modified and the updated version is depicted in Figure 5.14.

NTU RGB+D dataset requires some further steps before running the action recognition algorithm, and they consist in the rotation of the skeletons to compensate the view point, the removal of noisy data, and the identification of the main skeleton if more than one is present. The skeleton rotation is implemented following the approach described in [97], the x-axis is aligned to the vector connecting the shoulders and the y-axis is aligned to the "spine vector", going from spine base to spine joints. The effect of the rotation on a skeleton is shown in Figure 5.15, where the original and rotated skeletons related to the same body performing the *brushing teeth* action are shown. Figures 5.15(a-b) show original and rotated skeletons captured from camera 1, related to the 45° view, while Figures 5.15(c-d) show the same skeletons captured from camera 3, which observes the front view. The detection and filtering of noisy skeleton data is necessary because the dataset may include spurious skeletons, for example detected from objects. A skeleton is marked as noisy if their joints

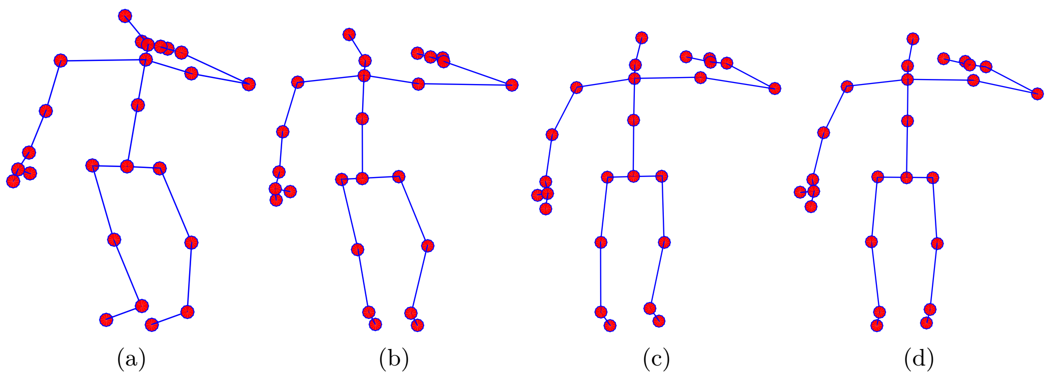


Figure 5.15.: Original and rotated skeletons related to the action *brushing teeth* captured from camera 1 (a-b) and camera 3 (c-d).

show a spread over x-axis higher than 0.8 of the one over y-axis. The measure of the spread is given by the standard deviation. This filtering process is applied with two different strategies, depending if it is considered in training phase or testing phase.

During training phase, the activity label is known, thus it is possible to know if the activity belongs to the group of actions (performed by one actor) or to the group of interactions (with two actors). If a sequence belonging to the former group contains more than one skeleton, the spread of skeleton joints over x and y axes is computed, and all the noisy skeletons are removed. In addition, if all the skeletons are noisy, the frame is removed and the sequence is shortened while if none of them is noisy, only the first one is kept. If a sequence belonging to the second group (interactions) contains more than two skeletons, the filtering technique based on the x and y spread is implemented to find the noisy ones. If none of the skeleton verifies the condition, only the first two skeletons are kept.

During testing phase the activity label is unknown, thus it is not possible to establish if a sequence is an action or interaction, and how many skeletons have to be present in the sequence. However, the coordinates of each skeleton are processed to establish if it is a noisy one or not. All the skeletons marked as noisy are removed and the assumption that the maximum number of skeletons in a sequence is two is made. If a sequence contains more than two skeletons, only the first two are kept even if none of them is noisy. If there are two skeletons in a sequence, the main one has to be calculated to extract the center-of-mass. The main skeleton is defined as the one featuring the highest amount of body motion. The body motion is given by the sum of position displacements of each joint from one frame to the next one.

Experimental tests have been executed setting $C = 1$ for the linear SVM, consid-

Table 5.12.: Accuracy (%) obtained by the proposed method on temporal pyramid of key poses (TPKP), with different clusters (K) and different levels of the temporal pyramid (L).

	cross-subject			cross-view		
K	$L = 2$	$L = 3$	$L = 4$	$L = 2$	$L = 3$	$L = 4$
4	41.8	45.2	43.8	44.3	49.1	50.1
8	44.5	48.1	45.0	49.3	53.0	52.5
16	46.4	48.7	45.1	52.3	56.3	54.9
32	48.1	**48.9**	46.8	55.1	57.2	56.2
64	47.7	48.3	–	55.3	**57.7**	–

ering a number of levels (L) going from 2 to 4 for the temporal pyramid, and the a number of key poses which is the same for each action, setting $K = K_1 = K_2 = \cdots = K_M$. The interval $K = [4, 8, 16, 32, 64]$ has been considered for the number of key poses and the accuracy has been evaluated for cross-subject and cross-view tests described in Subsection 2.2.3. Results are shown in Table 5.12, where it can be noticed that cross-subject evaluation is more challenging than the cross-view one, and also that better results are achieved with a large number of clusters.

The best accuracy for the cross-subject test (48.9%) has been obtained with $L = 3$ and $K = 32$, which means a total number of key poses of 1920 for the entire dataset constituted by 60 activities. With the cross-subject and $L = 3$ the accuracy does not decrease too much using a lower number of key poses: considering $K = 8$ the obtained performance is 48.1% and the minimum value is 45.2% with $K = 4$. The inclusion of 2 levels in the temporal pyramid allows to achieve comparable results if a large number of key poses are considered ($K = 32$ or $K = 64$), while a larger difference in the accuracy values is obtained when the number of clusters is reduced ($K = 4$ or $K = 8$). Choosing 4 levels for the temporal pyramid leads to a more complex system and does not improve the results with respect to $L = 3$. The configuration $L = 4$ and $K = 64$ has not been evaluated due to lack of memory.

Considering cross-view test, the best performance (57.7%) is achieved with $L = 3$ and $K = 64$. In this evaluation scheme, decreasing the number of key poses affects more the performance than in the cross-subject configuration, with a reduction of 8.6% using $K = 4$. The adoption of a temporal pyramid with 2 levels leads to worse performance, especially with reduced number of key poses, and the minimum

Table 5.13.: Comparison of different methods evaluated on NTU RGB+D dataset in terms of recognition accuracy (%). Results are ordered considering the accuracy of the cross-subject test.

Method	cross-subject	cross-view
Depth-based		
Oreifej and Liu [66]	30.56	7.26
Yang and Tian [64]	31.82	13.61
Ohn-Bar and Trivedi [172]	32.24	22.27
Skeleton-based		
Evangelidis et al. [163]	38.62	41.36
TPKP	48.9	57.7
Vemulapalli et al. [94]	50.08	52.76
Hu et al. [173]	60.23	65.22
Deep neural networks		
Du et al. [167]	59.07	63.97
Shahroudy et al. [97]	62.93	70.27
Liu et al. [174]	**69.2**	**77.7**

accuracy (44.3%) is represented by the simplest configuration, with $L = 2$ and $K = 4$. On the other hand, the consideration of $L = 4$ gives better results with a reduced number of key poses, while if K increases, the adoption of $L = 3$ achieves better results. Also for the cross-view evaluation scheme, the configuration $L = 4$ and $K = 64$ has not been considered.

The performance obtained by the method based on temporal pyramid of key poses is compared to other works proposed in literature in Table 5.13. The TPKP method achieves mid state-of-the-art results outperforming the three considered depth-based methods and only some of the skeleton-based approaches: Evangelidis et al. [163] in cross-subject and cross-view evaluation schemes and Vemulapalli et al. [94] only in cross-view. However, due to the complexity of the dataset, TPKP algorithm is not able to reach results achieved by data-hungry methods based on deep neural networks [97, 167, 174].

5.3. Conclusion

The algorithm based on AFV has been able to overcome the state-of-the-art results on KARD dataset. In particular, in the most challenging scenario which is the leave-one-actor-out cross validation test with the entire dataset, the AFV method achieved a precision of 95.1% and a recall of 95%. A gap of about 10% is obtained, with respect to the algorithm proposed by Gaglio et al. [90] who released the dataset. The same evaluation setting has been adopted for CAD-60 dataset, where the AFV algorithm has achieved a precision of 93.9% and a recall of 93.5%, comparable to the state-of-the-art. The results obtained on UTKinect are only 2% lower than the state-of-the-art, which is 97.1%. A mid state-of-the-art accuracy has been achieved on Florence3D, which is limited by high inter-class similarity and intra-class variability, and MSR Action3D, where the main challenge is again the similarity of different gestures.

The TPKP algorithm has been proposed to improve the results on the most challenging dataset, i.e. MSR Action3D. The adoption of the bag of key poses model with the temporal pyramid, and the parameters optimization through evolutionary computation methods, allowed to reach results close to the state-of-the-art on the cross-subject test. In particular, the accuracy of TPKP algorithm has achieved 95.14% against the best result claimed by Shahroudy et al. [170] which exploited also depth data. The TPKP algorithm, with some minor changes, has been tested also on the large-scale NTU RGB+D dataset, constituted by 60 actions performed by 40 actors for a total number of 56880 sequences. In this case, even if the TPKP method was not able to reach the state-of-the-art represented by data-hungry deep learning methods, it can be seen as an alternative approach of limited complexity, among the skeleton-based ones.

Chapter 6. Conclusion

In this thesis, different algorithms and applications based on RGB-D devices for smart environments have been proposed. In particular, the implemented algorithms can be used for mobility assessment and for human action recognition. This chapter initially includes a discussion summarizing the main results achieved in this thesis. Then, the contribution of the thesis is highlighted and the open issues and future research directions are presented. Finally, a list of publications is provided.

6.1. Discussion

After the introduction, a review of the literature on the two main topics addressed in this thesis, i.e. mobility assessment and human action recognition has been provided. Chapter 2 included an overview of different sensors and technologies adopted for mobility assessment, which is often considered for the evaluation of the fall risk, and human action recognition. A more detailed review on methods based on RGB-D sensors has been included for both topics, and the most used datasets have been discussed.

A tool for mobility assessment has been proposed in Chapter 3. The algorithm considers depth frames extracted from Kinect working in side view, i.e. on the sagittal plane, and calculates the coordinates of six joints: head, shoulder, elbow, hip, knee and ankle. The validation of the algorithm has been carried out with two different procedures. The former one considered infrared sticky markers attached on the body surface as reference and the error in the estimation of the joint positions was the metric considered for evaluation. The proposed algorithm is characterized by lower values in terms of average and standard deviation of the error over 18 repetitions of the test, if compared with the markerless estimation algorithms provided by Microsoft SDK and OpenNI library. A further validation step has been performed considering a marker-based stereometric system as a reference. Two fundamental angles for the evaluation of the sit-to-stand movement have been computed from the joint coordinates provided by the four systems. The absolute difference between the angle trajectories given by each markerless algorithm and the stereometric system is considered as the error metric. Four different repetitions of the test performed by two people resulted in a lower average and standard deviation of the error given by the proposed algorithm in almost all the cases for the estimation of fundamental angles.

In order to increase the number of parameters to perform a mobility assessment test, the RGB-D sensor can be used jointly with an IMU. A synchronization procedure to correctly associate samples from different sources has been proposed in Chapter 4. The delay times present between the acquisition of the frame from Kinect and its availability on the machine have been estimated for each stream of Kinect v1 and v2. The synchronization procedure is based on the compensation of delay times regarding Kinect data and in the linear regression of timestamps associated to acceleration samples transmitted through Bluetooth interface. The joint use of Kinect and acceleration sensor allows to extract a set of objective indices, such as timing, cadence and angle values, that can be useful for fall risk and mobility assessment. The combination of RGB-D and acceleration devices has been also adopted for fall detection. It allows to design simpler algorithms than the ones required if data are used independently. In fact, the acceleration data simplify the process of distinguishing a fall from lying on the ground action while the Kinect helps to separate actions generating a similar acceleration peak, such as sitting on a chair or falling.

The other main topic addressed in this thesis is represented by human action recognition, described in Chapter 5. An algorithm based on skeleton joints and Activity Feature Vectors (AFV) has been initially proposed as a solution that can achieve state-of-the-art results on KARD and CAD-60 datasets. In particular, considering the leave-one-actor-out cross-validation scenario, the algorithm achieves about 95% of precision and recall on KARD dataset while it reaches 93.9% and 93.5% respectively on CAD-60. The evaluation of this algorithm on Florence3D, UTKinect and MSR Action3D datasets has provided average state-of-the-art results, since the AFV algorithm has been overcome by other methods exploiting techniques for temporal alignment of the sequences, or a combination of several machine learning methods. The results on MSR Action3D dataset have been improved considering a different approach, based on Temporal Pyramid of Key Poses (TPKP). This method adopts the bag of key poses model and represents the sequence with the histograms of key poses calculated at each level of a temporal pyramid. The optimization of parameters with evolutionary computation allows to improve the results on MSR Action3D achieving an accuracy on the cross-subject test of 95.14%, which is close to the state-of-the-art. The TPKP method has been applied with a few changes on NTU RGB+D, a large-scale dataset recently released. It has achieved mid state-of-the-art results since better performance is given by data-hungry methods based on deep neural networks. However, TPKP algorithm can be considered as an alternative approach among the skeleton-based solutions.

6.2. Contributions

The contributions of this work can be summarized as follows:

- design and implementation of an algorithm for the extraction of skeleton joints from a depth frame. The algorithm works in side view and can be used to calculate objective indices during the sit-to-stand test. The validation of the proposed solutions with a stereometric system gave better results in terms of estimation of fundamental angles, if compared with other markerless joint estimation algorithms.

  Differently from other instrumented mobility assessment tests, which are based on skeleton coordinates provided by RGB-D sensors or on some information provided by the human shape, the approach proposed in Chapter 3 estimates the coordinates of six joints from the depth map. Since it works on the sagittal plane, it can be adopted when there are some limitations given by the environment or when the analysis from this point of view is of particular interest.

  This work has been published in [124, 125, 126].

- Development of a synchronization strategy to associate samples from a vision-based system (Kinect v1 and v2) and an IMU equipped with Bluetooth communication interface. The joint use of Kinect v2 and has been adopted to obtain indices from the execution of the TUG test and to implement fall detection algorithms.

  The main contribution is represented by the characterization of delay times affecting the different streams of data provided by Microsoft Kinect sensor. Even if the described synchronization strategy has been developed to associate a frame from a vision-based sensor to an acceleration sample, the approach can be adopted as it is to associate data coming from a Bluetooth channel, assuming that the time required to acquire data at the node is negligible. The delay times of data generated by Kinect can be also used to synchronize these data to others sent to the same machine through different communication protocols, if the delay introduced by the transmission process is known.

In order to demonstrate the feasibility of the data fusion method, two relevant applications in AAL domain have been considered: TUG and fall detection. Regarding the former one, to the best of the author's knowledge, this work has been the first one to propose the joint use of Kinect and IMU to instrument this test. Regarding the fall detection application, even if there are previous works considering the data fusion of vision-based and inertial data [175], this is the first one that addressed the synchronization issue.

This work has been published in [139, 140, 141].

- Proposal and validation of different HAR algorithms. The algorithm based on activity feature vectors (AFV) can achieve state-of-the-art results on KARD and CAD-60 datasets. The adoption of the bag of key poses model and the representation of a sequence as a temporal pyramid of key poses (TPKP) has allowed to achieve state-of-the-art results on the well-known MSR Action3D dataset.

  The algorithm based on AFV, despite its simplicity if compared to other fancy methods available in literature, is able to achieve far better results. A further improvement has been obtained with the combination of the bag of key poses method with a temporal pyramid. The TPKP algorithm has reached better results than previous methods exploiting the bag of key poses on MSR Action3D dataset.

  This work has been published in [149, 150, 151].

- Implementation of two recording tools to capture and store data from Kinect v1 and v2, described in Chapter A. These software have been used to capture five datasets that have been released to the research community.

6.3. Open issues and future works

This section analyzes the open issues in the proposed algorithms and applications identifying some future research directions.

The algorithm for joint extraction working in side view has been developed specifically for the sit-to-stand test followed by some steps. It requires a fixed background and this limitation, that can be easily satisfied only in a controlled environment, can be overcome considering the online update of the background, for example with an approach based on mixture of Gaussians. The robustness of the joint estimation algorithm could be improved with the adoption of a machine learning algorithm similar to the one implemented by Microsoft for the estimation of skeleton joints on the frontal plane, based on Random Forests.

The Timed Up and Go test instrumented with Kinect v2 and an IMU should be validated with a system representing the ground truth. As it has been done with the algorithm working in side view, a possible solution could be represented by a marker-based stereometric system providing the real coordinates of skeleton joints. Timing information and the length of the steps could be extracted from markers' trajectories and compared to the values calculated from kinect data. After the validation and tuning of the algorithms to extract objective indices, elderly people of different categories (healthy subjects and people at risk of falling) should be recruited to know what are their performance. The final aim of this work should be an automatic classification of the subject after the execution of the test, and the output should be an indicator stating if the subject is at risk of falling or not. This system could support the final decision of the clinicians.

The collection of a larger dataset, involving more actions and more actors, could help the development of a more robust fall detection algorithm. A machine learning method could be applied to distinguish a fall from an ADL using features extracted from skeleton joints and acceleration samples. The information fusion could be implemented for example at feature extraction level by using feature samples from all sensors at a centralized classifier, or at decision level by combining the decisions of independent classifiers based on data from each sensor.

The algorithm for human action recognition based on temporal pyramid of key poses generates the codebook using the well known clustering algorithm k-means. The adoption of a clustering algorithm which is aware of the class to which a sequence belongs may improve the separation of key poses associated to different classes in the codebook. A greater distance among key poses of different classes would probably increase the classification accuracy. Furthermore, even if it may be not usual in this field, the implementation of a statistical significance test could be performed to compare the different techniques evaluated. In order to improve the performance on large-scale datasets, such as NTU RGB+D, the development of a deep learning algorithm, capable to handle many different actions performed by different actors, could increase the recognition accuracy. Another aspect that can be addressed in the future is the application of an action recognition algorithm on real world scenario. In this case, two different things should be considered. Firstly, the developed algorithms work on sequences of frames previously captured. A segmentation algorithm has to be implemented to extract a sequence of frames representing an action from the stream of data acquired by the sensor. Secondly, working with real world data could bring to the evaluation of sequences that do not represent any action of the dataset. An unknown class should be included to label sequences that cannot be associated to classes used to train the system.

6.4. Publications

A list of scientific papers published is provided. The list is divided in three categories: journal papers, conference and workshop papers, and book chapters, some of which have been also presented in conferences or workshops.

Journal papers

- E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, “A Human Activity Recognition System Using Skeleton Data from RGBD Sensors,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 4351435, 14 pages, 2016. doi:10.1155/2016/4351435.

- E. Cippitelli, S. Gasparrini, S. Spinsante, E. Gambi, “Kinect as a Tool for Gait Analysis: Validation of a Real Time Joints Extraction Algorithm Working in Side View,” Sensors 2015, 15(1), 1417-1434. doi:10.3390/s150101417.

- L. Montanini, E. Cippitelli, E. Gambi, S. Spinsante, “Low Complexity Head Tracking on Portable Android Devices for Real Time Message Composition,” WILEY Journal on Multimodal User Interfaces, 2015. doi: 10.1007/s12193-015-0174-7.

- S. Gasparrini, E. Cippitelli, S. Spinsante, E. Gambi, “A Depth-Based Fall Detection System Using a Kinect Sensor,” Sensors 2014, 14(2), 2756-2775. doi:10.3390/s140202756.

Conference/workshop papers

- E. Cippitelli, S. Gasparrini, E. Gambi and S. Spinsante, “An Integrated Approach to Fall Detection and Fall Risk Estimation Based on RGB-Depth and Inertial Sensors,” International conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, special track on Emergent Technologies for Ambient Assisted Living (ETAAL) 2016, Vila Real (PT), December 1-3, 2016.

- E. Cippitelli, E. Gambi, S. Spinsante, and F. Florez-Revuelta, “Evaluation of a skeleton-based method for human activity recognition on a large-scale RGB-D dataset,” 2nd IET International Conference on Technologies for Active and Assisted Living (TechAAL) 2016, London (UK), October 24-25, 2016.

- E. Cippitelli, S. Gasparrini, E. Gambi and S. Spinsante, “Unobtrusive Intake Actions Monitoring Through RGB and Depth Information Fusion,” IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP) 2016, Cluj-Napoca (RO), September 8-10, 2016.

- S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante and F. Florez-Revuelta, “Performance Analysis of Self-Organising Neural Networks Tracking Algorithms for Intake Monitoring Using Kinect,” IET International Conference on Technologies for Active and Assisted Living (TechAAL) 2015, Kingston upon Thames (UK), November 5, 2015.

- E. Cippitelli, S. Gasparrini, S. Spinsante, E. Gambi, F. Verdini, L. Burattini, F. Di Nardo and S. Fioretti, “Validation of an Optimized Algorithm to Use Kinect in a Non-Structured Environment for Sit-to-Stand Analysis,” 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan (IT), August 25-29, 2015.

- E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, J. Wåhslén, I. Orhan, and T. Lindh, “Time Synchronization and Data Fusion for RGB-Depth Cameras and Wearable Inertial Sensors in AAL Applications,” IEEE ICC 2015 - Workshop on ICT-Enabled Services and Technologies for eHealth and Ambient Assisted Living, London (UK), June 8-12, 2015.

- E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “Depth Stream Compression for Enhanced Real Time Fall Detection by Multiple Sensors,” IEEE

International Conference on Consumer Electronics (ICCE-Berlin) 2014, Berlin (DE), September 7-10, 2014.

- E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “A Depth-Based Joints Estimation Algorithm for Get Up and Go Test Using Kinect,” IEEE International Conference on Consumer Electronics (ICCE) 2014, Las Vegas (US), January 10-13, 2014.
- L. Montanini, E. Cippitelli, E. Gambi, S. Spinsante, “Real time message composition through head movements on portable Android devices,” IEEE International Conference on Consumer Electronics (ICCE) 2014, Las Vegas (US), January 10-13, 2014.

Book chapters

- E. Cippitelli, E. Gambi, S. Spinsante, and F. Florez-Revuelta, “Human Action Recognition Based on Temporal Pyramid of Key Poses Using RGB-D Sensors,” Advanced Concepts for Intelligent Vision Systems, Springer International Publishing, 2016. 510-521, doi:10.1007/978-3-319-48680-2_45.
  Paper presented at ACIVS 2016, Lecce (IT), October 24th-27th, 2016. **Best Student Paper Award**.
- S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante, J. Wåhslén, I. Orhan and T. Lindh, “Proposal and Experimental Evaluation of Fall Detection Solution Based on Wearable and Depth Data Fusion,” ICT Innovations 2015, Springer International Publishing, 2016. 99-108, doi:10.1007/978-3-319-25733-4_11.
  Paper presented at ICT Innovations 2015, Workshop ELEMENT 2015, Orhid (FYROM), October 1-4, 2015.
- S. Gasparrini, E. Cippitelli, S. Spinsante, E. Gambi, “Depth Cameras in AAL Environments: Technology and Real-world Applications,” Chapter 2 in Theng, Lau Bee. “Assistive Technologies for Physical and Cognitive Disabilities.” IGI Global, 2015. 1-300. Web. 19 Dec. 2014. doi:10.4018/978-1-4666-7373-1.
- S. Spinsante, E. Cippitelli, A. De Santis, E. Gambi, S. Gasparrini, L. Montanini, and L. Raffaeli, “Multimodal Interaction in a Elderly-Friendly Smart Home: a Case Study,” Mobile Networks and Management. Springer International Publishing, 2015, 373-386, doi:10.1007/978-3-319-16292-8_27.
  Paper presented at ELEMENT Workshop, 6th International Conference, MONAMI 2014, WÃijrzburg (DE), 24 September, 2014. **Best Paper Award**.

- E. Cippitelli, S. Gasparrini, A. De Santis, L. Montanini, L. Raffaeli, E. Gambi, and S. Spinsante, “Comparison of RGB-D Mapping Solutions for Application to Food Intake Monitoring,” Ambient Assisted Living: Italian Forum 2014. Springer International Publishing, 2015, 295-305, doi:10.1007/978-3-319-18374-9_28.
  Paper presented at 5th Italian Forum on Ambient Assisted Living (foritAAL) 2014, Catania (IT), September 2-5, 2014.

- E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, “Quality of Kinect Depth Information for Passive Posture Monitoring,” Ambient Assisted Living: Italian Forum 2013. Springer International Publishing, 2014, 107-116, doi: 10.1007/978-3-319-01119-6_11.
  Paper presented at 4th Italian Forum on Ambient Assisted Living (foritAAL) 2013, Ancona (IT), October 23-25, 2013.

Appendices

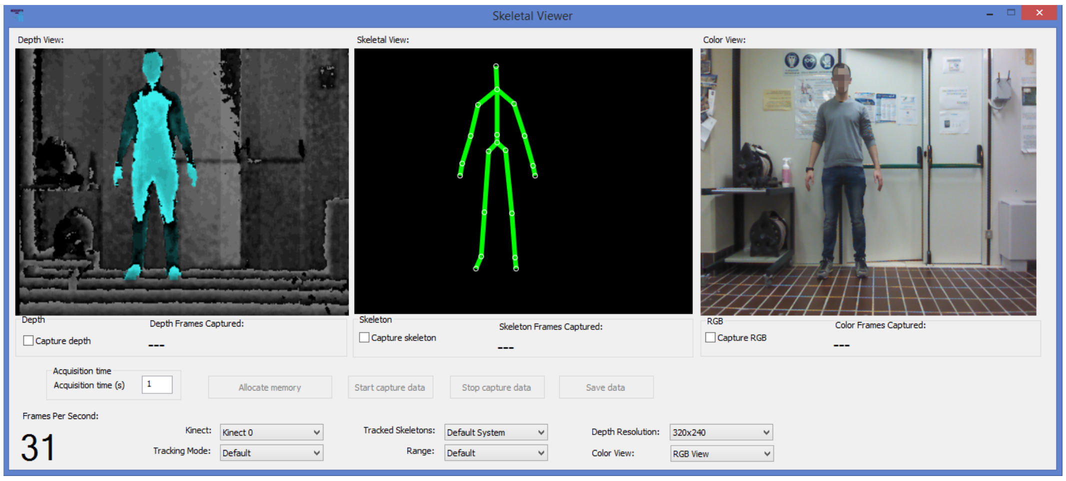
Appendix A.

A.1. Kinect v1 capture software

A software called *Skeletal viewer* has been developed to capture and store data provided by Kinect v1 sensor. This tool has been used to capture data required for offline processing, and also to provide some datasets to the community [146]. The software has been developed in C++ and it exploits the Microsoft SDK to enable the communication with the sensor. Since it is not possible to retrieve frames and to write them on-the-fly on a Hard Disk Drive (HDD), the software acquires data from Kinect and temporally stores data in a buffer in RAM memory. The data are then written on the hard disk at the end of the capturing process. Since *Skeletal viewer* exploits the RAM memory to memorize data when they are captured by the sensor, it can store all the frames, without the risk of losing frames due to limitations in the speed of writing buffer of the disk. On the other hand, the main drawback is that the maximum amount of data that can be captured is related to the RAM memory of the machine used.

The Graphical User Interface (GUI) is shown in Figure A.1, where it can be noticed that three windows are shown. Starting from the left, the first one is related to depth data, and each pixel is associated to a grayscale value except for the pixels related to human body that take a specific color, the second one is for the human skeleton while the last one is for RGB video. In particular, it is possible to export:

- RGB: each frame is stored as an image with resolution 640×480 in the bitmap format. A file related to RGB stream is named as `FrameRGB_ID.bmp`, where `ID` represents the identifier of the frame within a sequence.
- Skeleton: all the skeleton frames are stored in two files `.txt` and two files `.csv`, where the coordinates of skeleton joints in depth frame and real world are contained.
- Depth: each frame is stored as a binary vector with a resolution of 320×240 or



Acquisition time
Acquisition time (s) 1

Frames Per Second: 31

Kinect: Kinect 0
Tracking Mode: Default
Tracked Skeletons: Default System
Range: Default
Depth Resolution: 320x240
Color View: RGB View

Figure A.1.: GUI of *Skeletal viewer*, the software developed to capture data from Kinect v1.

640 × 480. The files related to depth are named as `Filedepth_ID.bin`, where `ID` has the same meaning of the RGB data.

- Timing information: a timestamp related to each frame captured is stored for synchronization purposes. Three different files named `RGBTime.csv`, `DepthTime.csv`, `SkelTime.csv` contain timestamps for RGB, depth and skeleton frames respectively.

Since the infrared data is considered as a specific configuration of the color stream, it is possible to capture alternatively IR or RGB frames. The selection is performed using the combo box identified by the label *Color View*.

This procedure can be followed to use the tool:

1. select the streams that have to be captured using the check boxes below each window.

2. Insert the amount of time (in seconds) that have to be captured and click the button *Allocate memory* which creates the data structure in RAM memory to temporally store the data. The software checks the amount of available RAM and, if necessary, reduces the maximum capture time to allow data acquisition.

3. Click the button *Start capture data* to begin the data acquisition process; the counters below the window of each selected stream are incremented every time a new frame is captured.

4. The acquisition process can be stopped using the button *Stop capture data* or it automatically ends when the maximum acquisition time is reached.

5. Store acquired data in the hard disk using the button *Save data*. Frames are stored at the path `C:\Saved_Data\Data_v1`, where a different folder is created for each selected stream.

The maximum acquisition time depends on the amount of available RAM and on the type of stream selected for acquisition process. In particular, the skeleton stream provides only the 3D skeleton coordinates of 20 joints for each frame, so the amount of required space is quite low. For depth and RGB frames it is possible to define as p_{row} and p_{col} respectively the number of rows and columns in one frame, with p_{bytes} the number of bytes required to represent a pixel and with fps the number of frames per second. The following simple equation can be used to calculate the amount of bytes for 1 second of data:

$$data_{1sec} = p_{row} \cdot p_{col} \cdot p_{bytes} \cdot fps \tag{A.1}$$

Each pixel of a depth frame requires $p_{bytes} = 2$, considering a resolution of 320×240 and $fps = 30$, from equation A.1 follows that it requires 4 608 000 bytes for 1 second of data. Equation (A.1) can be used also for RGB frames, considering $p_{col} = 640$, $p_{row} = 480$ and $p_{bytes} = 4$, due to the presence of the *alpha* channel. Thus, 1 second of RGB video requires 36 864 000 bytes. Considering these figures, it is possible to conclude that RGB data requires a space which is 8 times bigger than depth information, and it is the stream that makes the difference in the maximum amount of acquirable data.

The configurations listed in Table A.1 have been tested on a machine with 16 GB of RAM and a standard HDD. Considering the amount of time required by the operating system and leaving some free RAM for other programs, it is possible to acquire 260 seconds of RGB and depth data, with an amount of occupied RAM of about 11 GB. Considering only RGB frames the maximum capture time is a bit lower than 5 minutes while, capturing only depth data, the acquisition of 15 minutes of data requires only about 4 GB of RAM, and even more data could be considered.

In order to capture more than 5 minutes of data with RGB and depth information, it is necessary to adopt another approach and to use a Solid State Drive (SSD). The tool named *Kinect Stream Saver Application* [176] starts the writing process during the acquisition process. If the hard disk exploits the magnetic technology, the writing speed is not enough and the software crashes after around 6 minutes in the used

Table A.1.: Performance obtained by *Skeletal viewer* on a laptop with 16 GB of RAM and standard HDD considering different streams.

	RGB	Depth	RGB + Depth
RAM used [GB]	10.3	4.1	10.7
Capture time [s]	280	900	260
Save time [s]	300	120	300

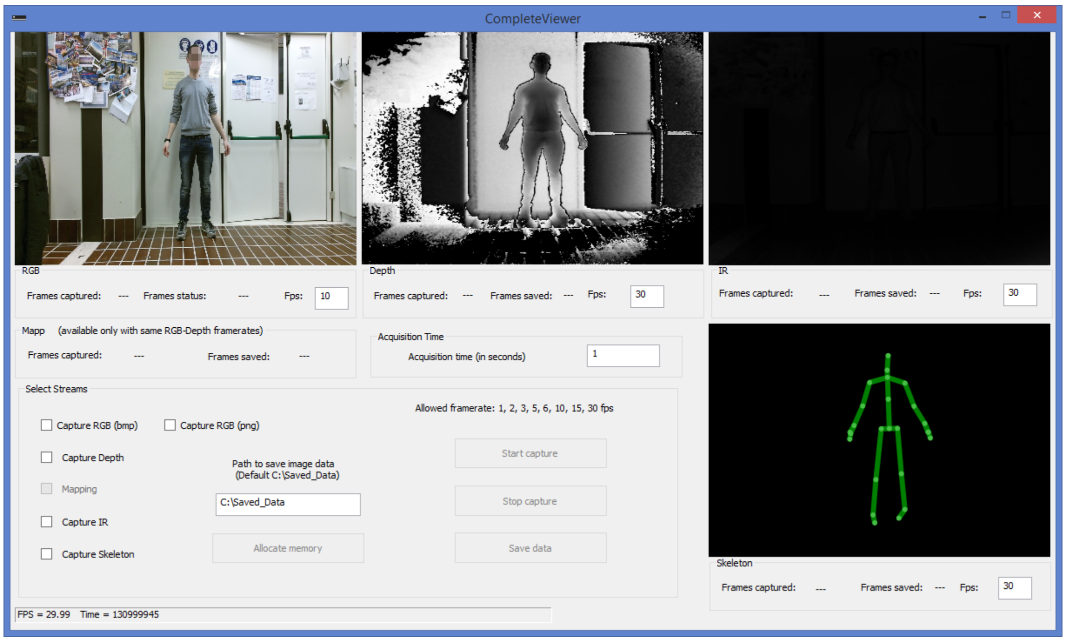
laptop with 16 GB of RAM. On the other hand, the adoption of a SSD allows to write data on the disk at real time without losing any frame of RGB and depth data.

The *Skeletal viewer* tool has been used to capture data related to the side-view mobility assessment analysis described in Chapter 3, and also to acquire and share with the research community some RGB-D datasets, available at [146]. Such datasets acquired with Kinect v1 are:

- TST Fall detection v1, which provides depth frames at a resolution of 320×240 of people simulating falls. Data have been captured in top-view configuration and four volunteers have been considered in the acquisition process, having a total number of 20 tests. The dataset can be split in two groups of 10 tests each: Group A, featuring multiple people walking in the area of interest and interacting with objects, and Group B, where one person is present in the area of interest and falls are simulated.
  This dataset has been used in [177].

- TST Intake Monitoring v1, a dataset of 48 sequences providing food and drink intake movements performed by 35 young people. The depth frames have been captured at a resolution of 320×240 considering the top-view configuration. This dataset has been used in [178].

- TST Intake Monitoring v2, a dataset acquired with the same aim of the previous one, providing 60 sequences performed by 20 people for 3 times. The three repetitions are related to different movements describing food and drink intake actions, such as drink snack, eat a soup or use knife and fork.



RGB — Frames captured: --- Frames status: --- Fps: 10

Depth — Frames captured: --- Frames saved: --- Fps: 30

IR — Frames captured: --- Frames saved: --- Fps: 30

Mapp — Frames captured: --- Frames saved: ---

Acquisition time (in seconds) 1

C:\Saved_Data

Skeleton — Frames captured: --- Frames saved: --- Fps: 30

FPS = 29.99 Time = 130999945

Figure A.2.: GUI of *Complete viewer*, the software developed to capture data from Kinect v2.

A.2. Kinect v2 capture software

The tool named *Complete viewer* has been developed to capture and store data from Kinect v2. With the latest version of Kinect sensor, the problem of capturing data and storing them in the disk is even more complex because the resolution of RGB frames (1920×1080) is much higher than the resolution of the previous version. Due to the large amount of data required by the RGB frames, it is possible to select two different formats for color images: the uncompressed bitmap format, and the compressed png format. The method `imwrite` provided by OpenCV library [179] has been used to compress and store the data using png format, obtaining an image file with a weight of 3 MB which is much less than the memory requirement of the bitmap file, i.e. about 8 MB.

The same approach of storing temporally the data in the RAM memory and then writing them on the disk at the end of acquisition process has been implemented. The possibility to choose the frame rate at which the data are captured has been also included, in order to further reduce the amount of data to be stored.

The GUI of *Complete viewer* is shown in Figure A.2, where four different windows, each of which related to one of the streams, can be noticed. In details, the following data can be captured:

- RGB: each frame is stored at the resolution of 1920×1080 and can be encoded with two different formats, the uncompressed bitmap format and the png format, which is compressed.

- Depth: depth frames are acquired at the resolution of 512×424 and they are stored with the same approach of *Skeletal viewer*, i.e. using a binary vector.

- Infrared: each frame has the same resolution of the depth stream, and it is stored as a bitmap image.

- Skeleton: the skeleton frames of a sequence are stored in two `.txt` files and two `.csv` files, containing the joints in depth coordinate space and world coordinate space.

- Mapping: the mapping information is represented by a matrix of 1024×424 which allows to find a correspondence between RGB and depth information.

- Timing information: a timestamp related to each frame is stored for synchronization. It is possible to have timestamps from Kinect library and from *QueryPerformanceCounter* (QPC) [143] stored in `ColorTime.csv`, `DepthTime.csv`, `BodyTime.csv`, `InfraredTime.csv`.

A user can interact with *Complete viewer* with the following steps:

1. select the streams that have to be captured; for RGB stream it is possible to choose the format to store the data (bmp or png), and the mapping stream can be saved only if the RGB and depth streams are enabled and if the frame rate is equal.

2. Choose the path to save data, which is `C:\Saved_Data` by default. Following the same structure for Kinect v1, a sub-folder called `Data_v2` is created, and the files belonging to different streams are stored.

3. Select the frame rate for each stream, using the textboxes labeled as “Fps” below each window; by default, 10 fps are used for RGB frames and 30 fps for the other streams. Only a selected set of frame rates (submultiples of 30) is allowed.

Table A.2.: Performance obtained by *Complete viewer* on a laptop with 16 GB of RAM and standard HDD considering different streams.

	RGB	Depth	RGB + Depth
RAM used [GB]	10	10.6	9.5
Capture time [s]	40	818	36
Save time [s]	146 (bmp) 1200 (png)	180	260 (bmp) 1200 (png)

4. Insert the maximum acquisition time, specified in seconds. The data structure in RAM memory to store the data is created by clicking the button *Allocate memory*. As implemented in *Skeletal viewer*, the software checks the amount of available RAM and automatically limits the capture time to allow data acquisition.

5. Start the acquisition process with the button *Start capture*.

6. Stop the acquisition process with *Stop capture* or wait for the reaching of the maximum acquisition time.

7. The data can be written on the disk with the button *Save data*.

Table A.2 shows the performance obtained considering a laptop with 16 GB of RAM and Intel i7 processor. The maximum capture time is quite low if the acquisition of RGB frames is enabled. In particular, by applying equation (A.1) with the resolution of Kinect v2 frames, i.e. 1920×1080 and 512×424 for RGB and depth respectively, the maximum capture time is about 40 seconds with RGB, which is much lower than the maximum time when only depth data are captured (about 800 seconds). It can be also noticed that the time required to store the data is considerably different choosing png or bmp format. This is reasonable because part of the time is required to convert the data structure of a frame provided by Microsoft SDK to the one accepted by the OpenCV library, and some time is necessary for compression. However, it has been observed that the time required to save data on hard disk can be very different and it depends on the usage of the machine. Two versions of the software *Complete viewer* are available for download at [146]. The only difference is that one of those does not require OpenCV library to be executed, so RGB frames can be saved only in bmp format.

This tool has been modified to add a serial communication interface with the IMU equipped with the Bluetooth transceiver. It has been used for recording and to develop algorithms exploiting acceleration and RGB-D data, as described in Chapter 4. The following datasets, available at [146], have been collected with the modified *Complete viewer*:

- TST TUG, a dataset constituted by depth, skeleton and inertial data captured during the Timed Up and Go test. Movements have been acquired from 20 people performing the mobility assessment test 3 times.
  This dataset has been used in the work described in Section 4.2 and published in [140].

- TST Fall detection v2, which provides depth, skeleton and acceleration data captured during the simulation of different ADLs and falls. A group of 11 healthy volunteers have been recruited for the simulation of 8 activities (4 ADLs and 4 falls) for 3 times.
  This dataset has been used in the work described in Section 4.3 and published in [141].

Bibliography

[1] Department of Economic and Social Affairs - Population Division, “World population ageing 2015,” United Nations, Report, 2015.

[2] ——, “World population prospects 2015,” United Nations, Report, 2015.

[3] D. N. Monekosso, F. Florez-Revuelta, and P. Remagnino, “Guest editorial special issue on ambient-assisted living: Sensors, methods, and applications,” *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 545–549, Oct 2015.

[4] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos, “A survey on ambient intelligence in healthcare,” *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2470–2494, Dec 2013.

[5] F. Sadri, “Ambient intelligence: A survey,” *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp. 36:1–36:66, Oct 2011.

[6] A. A. Chaaraoui and F. Florez-Revuelta, “Technologies and applications for active and assisted living-current situation,” in *Active and Assisted Living: Technologies and Applications*, ser. Healthcare Technologies. Institution of Engineering and Technology, 2016, pp. 1–8.

[7] Mylife project, http://www.aal-europe.eu/projects/my-life/, accessed: 2017-02-12.

[8] Rosetta project, http://www.aal-europe.eu/projects/help/, accessed: 2017-02-12.

[9] Domeo project, http://www.aal-domeo.org/, accessed: 2017-02-12.

[10] Home4Dem project, http://home4dem.eu/, accessed: 2017-02-12.

[11] P. Rashidi and A. Mihailidis, “A survey on ambient-assisted living tools for older adults,” *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, May 2013.

[12] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, “Automatic fall monitoring: A review,” *Sensors*, vol. 14, no. 7, pp. 12 900–12 936, 2014.

[13] M. Terroso, N. Rosa, A. Torres Marques, and R. Simoes, “Physical consequences of falls in the elderly: a literature review from 1995 to 2010,” *European Review of Aging and Physical Activity*, vol. 11, no. 1, pp. 51–59, 2014.

[14] A. Muro-de-la Herran, B. Garcia-Zapirain, and A. Mendez-Zorrilla, “Gait Analysis Methods: An Overview of Wearable and Non-Wearable Systems, Highlighting Clinical Applications,” *Sensors*, vol. 14, no. 2, pp. 3362–3394, 2014.

[15] S. Mathias, U. Nayak, and B. Isaacs, “Balance in elderly patients: the “get-up and go” test,” *Archives of physical medicine and rehabilitation*, vol. 67, no. 6, pp. 387–389, June 1986.

[16] D. Podsiadlo and S. Richardson, “The timed “up & go”: a test of basic functional mobility for frail elderly persons,” *Journal of the American geriatrics Society*, vol. 39, no. 2, pp. 142–148, 1991.

[17] G. Sprint, D. J. Cook, and D. L. Weeks, “Toward automating clinical assessments: A survey of the timed up and go,” *IEEE Reviews in Biomedical Engineering*, vol. 8, pp. 64–77, 2015.

[18] B. R. Greene, A. O’Donovan, R. Romero-Ortuno, L. Cogan, C. N. Scanaill, and R. A. Kenny, “Quantitative falls risk assessment using the timed up and go test,” *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 12, pp. 2918–2926, Dec 2010.

[19] M. R. Narayanan, S. J. Redmond, M. E. Scalzi, S. R. Lord, B. G. Celler, and N. H. Lovell, “Longitudinal falls-risk estimation using triaxial accelerometry,” *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 3, pp. 534–541, March 2010.

[20] S. K. SankarPandi, S. Dlay, W. L. Woo, and M. Catt, “Predicting disability levels of community dwelling older individuals using single wrist mounted accelerometer,” in *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, June 2014, pp. 720–723.

[21] A. Salarian, F. B. Horak, C. Zampieri, P. Carlson-Kuhta, J. G. Nutt, and K. Aminian, “iTUG, a Sensitive and Reliable Measure of Mobility,” *IEEE*

Transactions on Neural Systems and Rehabilitation Engineering, vol. 18, no. 3, pp. 303–310, June 2010.

[22] L. Palmerini, S. Mellone, L. Rocchi, and L. Chiari, "Dimensionality reduction for the quantitative evaluation of a smartphone-based timed up and go test," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2011, pp. 7179–7182.

[23] M. Milosevic, E. Jovanov, and A. Milenković, "Quantifying timed-up-and-go test: A smartphone implementation," in *2013 IEEE International Conference on Body Sensor Networks*, May 2013, pp. 1–6.

[24] J. Silva and I. Sousa, "Instrumented timed up and go: Fall risk assessment based on inertial wearable sensors," in *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, May 2016, pp. 1–6.

[25] B. Williams, B. Allen, H. True, N. Fell, D. Levine, and M. Sartipi, "A real-time, mobile timed up and go system," in *2015 IEEE 12th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, June 2015, pp. 1–6.

[26] S. Demura and M. Uchiyama, "Proper assessment of the falling risk in the elderly by a physical mobility test with an obstacle," *The Tohoku Journal of Experimental Medicine*, vol. 212, no. 1, pp. 13–20, 2007.

[27] G. Baldewijns, G. Verheyden, B. Vanrumste, and T. Croonenborghs, "Validation of the kinect for gait analysis using the gaitrite walkway," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2014, pp. 5920–5923.

[28] Z. Skrba, B. O'Mullane, B. R. Greene, C. N. Scanaill, C. W. Fan, A. Quigley, and P. Nixon, "Objective real-time assessment of walking and turning in elderly adults," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Sept 2009, pp. 807–810.

[29] F. Wang, M. Skubic, C. Abbott, and J. M. Keller, "Quantitative analysis of 180 degree turns for fall risk assessment using video sensors," in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2011, pp. 7606–7609.

[30] O. Lohmann, T. Luhmann, and A. Hein, "Skeleton timed up and go," in *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, Oct 2012, pp. 1–5.

[31] N. Kitsunezaki, E. Adachi, T. Masuda, and J. i. Mizusawa, "Kinect applications for the physical rehabilitation," in *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium on*, May 2013, pp. 294–299.

[32] A. Hassani, A. Kubicki, V. Brost, and F. Yang, "Preliminary study on the design of a low-cost movement analysis system reliability measurement of timed up and go test," in *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, vol. 2, Jan 2014, pp. 662–667.

[33] ——, "Real-time 3d tug test movement analysis for balance assessment using microsoft kinect," in *Proceedings of the AI* IA 2014 Symposium on Artificial Intelligence for Ambient Assisted Living*, 2014.

[34] E. Gianaria, M. Grangetto, M. Roppolo, A. Mulasso, and E. Rabaglietti, "Kinect-based gait analysis for automatic frailty syndrome assessment," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 1314–1318.

[35] D. Leightley, M. H. Yap, J. Coulson, Y. Barnouin, and J. S. McPhee, "Benchmarking human motion analysis using kinect one: An open source dataset," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Dec 2015, pp. 1–7.

[36] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "A Survey of Online Activity Recognition Using Mobile Phones," *Sensors*, vol. 15, no. 1, pp. 2059–2085, 2015.

[37] O. D. Lara and M. A. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[38] R. Damaševičius, M. Vasiljevas, J. Šalkevičius, and M. Woźniak, "Human Activity Recognition in AAL Environments Using Random Projections," *Computational and Mathematical Methods in Medicine*, vol. 2016, 2016.

[39] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, “Physical human activity recognition using wearable sensors,” *Sensors*, vol. 15, no. 12, pp. 31 314–31 338, 2015.

[40] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, “Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, Jan 2006.

[41] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen, “Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers,” *Pattern Recognition Letters*, vol. 29, no. 16, pp. 2213–2220, 2008.

[42] L. Gao, A. Bourke, and J. Nelson, “Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems,” *Medical Engineering & Physics*, vol. 36, no. 6, pp. 779–785, 2014.

[43] Q. Ni, A. B. García Hernando, and I. P. de la Cruz, “The Elderly’s Independent Living in Smart Homes: A Characterization of Activities and Sensing Infrastructure Survey to Facilitate Services Development,” *Sensors*, vol. 15, no. 5, pp. 11 312–11 362, 2015.

[44] E. M. Tapia, S. S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” in *Pervasive Computing: Second International Conference, PERVASIVE 2004, Linz/Vienna, Austria, April 21-23, 2004. Proceedings*, A. Ferscha and F. Mattern, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 158–175.

[45] F. J. Ordóñez, P. de Toledo, and A. Sanchis, “Activity recognition using hybrid generative/discriminative models on home environments using binary sensors,” *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.

[46] D. H. Wilson and C. Atkeson, “Simultaneous Tracking and Activity Recognition (STAR) Using Many Anonymous, Binary Sensors,” in *Pervasive Computing: Third International Conference, PERVASIVE 2005, Munich, Germany, May 8-13, 2005. Proceedings*, H. W. Gellersen, R. Want, and A. Schmidt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 62–79.

[47] J. M. Sim, Y. Lee, and O. Kwon, “Acoustic sensor based recognition of human activity in everyday life for smart home services,” *International Journal of Distributed Sensor Networks*, vol. 2015, p. 24, 2015.

[48] S. Tremblay, D. Fortin-Simard, E. Blackburn-Verrault, S. Gaboury, and B. Bouchard, "Exploiting environment sounds for activity recognition in smart homes," in *2nd AAAI Workshop on Artificial Intelligence Applied to Assistive Technologies and Smart Environments (ATSE '15)*, January 2015.

[49] E. L. Salomons and P. J. M. Havinga, "A survey on the feasibility of sound classification on wireless sensor nodes," *Sensors*, vol. 15, no. 4, pp. 7462–7498, 2015.

[50] L. Vuegen, B. Van Den Broeck, P. Karsmakers, H. Van hamme, and B. Vanrumste, "Energy efficient monitoring of activities of daily living using wireless acoustic sensor networks in clean and noisy conditions," in *Signal Processing Conference (EUSIPCO), 2015 23rd European*, Aug 2015, pp. 449–453.

[51] L. Vuegen, B. Van Den Broeck, P. Karsmakers, H. Van Hamme, and B. Vanrumste, "Monitoring activities of daily living using wireless acoustic sensor networks in clean and noisy conditions," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug 2015, pp. 4966–4969.

[52] M. Salman Khan, M. Yu, P. Feng, L. Wang, and J. Chambers, "An unsupervised acoustic fall detection system using source separation for sound interference suppression," *Signal Processing*, vol. 110, pp. 199–210, 2015, machine learning and signal processing for human pose recovery and behavior analysis.

[53] Y. Li, Z. Zeng, M. Popescu, and K. C. Ho, "Acoustic fall detection using a circular microphone array," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, Aug 2010, pp. 2242–2245.

[54] S. Wang and G. Zhou, "A review on radio based activity recognition," *Digital Communications and Networks*, vol. 1, no. 1, pp. 20–29, 2015.

[55] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: Device-free location-oriented activity identification using fine-grained wifi signatures," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '14. New York, NY, USA: ACM, 2014, pp. 617–628.

[56] S. Shi, S. Sigg, and Y. Ji, "Joint localization and activity recognition from ambient fm broadcast signals," in *Proceedings of the 2013 ACM Conference*

on Pervasive and Ubiquitous Computing Adjunct Publication, ser. UbiComp '13 Adjunct. New York, NY, USA: ACM, 2013, pp. 521–530.

[57] B. Çağlıyan and S. Z. Gürbüz, "Micro-doppler-based human activity classification using the mote-scale bumblebee radar," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 10, pp. 2135–2139, Oct 2015.

[58] J. Aggarwal and Q. Cai, "Human motion analysis: A review," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 428–440, 1999.

[59] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, 2003.

[60] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, Nov 2008.

[61] A. A. Chaaraoui, P. Climent-Perez, and F. Florez-Revuelta, "A review on vision techniques applied to Human Behaviour Analysis for Ambient-Assisted Living," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 873–10 888, 2012.

[62] T. D'Orazio, R. Marani, V. Renò, and G. Cicirelli, "Recent trends in gesture recognition: how depth data has improved classical approaches," *Image and Vision Computing*, vol. 52, pp. 56–72, 2016.

[63] J. K. Aggarwal and L. Xia, "Human activity recognition from 3d data: A review," *Pattern Recognition Letters*, vol. 48, pp. 70–80, 2014.

[64] X. Yang and Y. Tian, "Super Normal Vector for Activity Recognition Using Depth Sequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[65] R. Slama, H. Wannous, and M. Daoudi, "Grassmannian Representation of Motion Depth for 3D Human Gesture and Action Recognition," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, Aug 2014, pp. 3499–3504.

[66] O. Oreifej and Z. Liu, "HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences," in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 716–723.

[67] H. Rahmani, A. Mahmood, D. Q Huynh, and A. Mian, “HOPC: Histogram of Oriented Principal Components of 3D Pointclouds for Action Recognition,” in *Computer Vision - ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8690, pp. 742–757.

[68] W. Li, Z. Zhang, and Z. Liu, “Action recognition based on a bag of 3D points,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2010, pp. 9–14.

[69] A. A. Chaaraoui, P. Climent-Perez, and F. Florez-Revuelta, “Silhouette-based human action recognition using sequences of key poses,” *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1799–1807, 2013.

[70] L. Xia and J. Aggarwal, “Spatio-temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 2834–2841.

[71] A. Vieira, E. Nascimento, G. Oliveira, Z. Liu, and M. Campos, “STOP: Space-Time Occupancy Patterns for 3D Action Recognition from Depth Map Sequences,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7441, pp. 252–259.

[72] G. Chen, D. Clarke, M. Giuliani, A. Gaschler, and A. Knoll, “Combining unsupervised learning and discrimination for 3d action recognition,” *Signal Processing*, vol. 110, pp. 67–81, 2015.

[73] S. Althloothi, M. H. Mahoor, X. Zhang, and R. M. Voyles, “Human activity recognition using multi-features and multiple kernel learning,” *Pattern Recognition*, vol. 47, no. 5, pp. 1800–1812, 2014.

[74] Y. Zhu, W. Chen, and G. Guo, “Fusing Spatiotemporal Features and Joints for 3D Action Recognition,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2013, pp. 486–491.

[75] I. Laptev and T. Lindeberg, “Velocity adaptation of space-time interest points,” in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 1, Aug 2004, pp. 52–56.

[76] G. Willems, T. Tuytelaars, and L. Van Gool, "An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector," in *Computer Vision - ECCV 2008*, ser. Lecture Notes in Computer Science, D. Forsyth, P. Torr, and A. Zisserman, Eds. Springer Berlin Heidelberg, 2008, vol. 5303, pp. 650–663.

[77] A. Kläser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *Proceedings of the British Machine Vision Conference (BMVC'08)*, 2008, pp. 995–1004.

[78] A. Chaaraoui, J. Padilla-Lopez, and F. Florez-Revuelta, "Fusion of Skeletal and Silhouette-Based Features for Human Action Recognition with RGB-D Devices," in *2013 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Dec 2013, pp. 91–97.

[79] J. Luo, W. Wang, and H. Qi, "Spatio-temporal feature extraction and representation for RGB-D human action recognition," *Pattern Recognition Letters*, p. doi: 10.1016/j.patrec.2014.03.024, 2014.

[80] Y. Zhu, W. Chen, and G. Guo, "Evaluating spatiotemporal interest point features for depth-based action recognition," *Image and Vision Computing*, vol. 32, no. 8, pp. 453–464, 2014.

[81] A.-A. Liu, W.-Z. Nie, Y.-T. Su, L. Ma, T. Hao, and Z.-X. Yang, "Coupled hidden conditional random fields for RGB-D human action recognition," *Signal Processing*, vol. 112, pp. 74–82, 2015.

[82] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Del Bimbo, "Space-Time Pose Representation for 3D Human Action Recognition," in *New Trends in Image Analysis and Processing - ICIAP 2013*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8158, pp. 456–464.

[83] L. Gan and F. Chen, "Human Action Recognition Using APJ3D and Random Forests," *Journal of Software*, vol. 8, no. 9, 2013.

[84] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, June 2012, pp. 1290–1297.

[85] L. Xia, C.-C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3D joints,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, pp. 20–27.

[86] J. Luo, W. Wang, and H. Qi, “Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps,” in *2013 IEEE International Conference on Computer Vision (ICCV)*, Dec 2013, pp. 1809–1816.

[87] C. Wang, Y. Wang, and A. Yuille, “An Approach to Pose-Based Action Recognition,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 915–922.

[88] X. Yang and Y. Tian, “Effective 3D action recognition using EigenJoints,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 2–11, 2014.

[89] A. Taha, H. H. Zayed, M. E. Khalifa, and E.-S. M. El-Horbaty, “Human Activity Recognition for Surveillance Applications,” in *ICIT 2015, The 7th International Conference on Information Technology*, 2015, pp. 577–586.

[90] S. Gaglio, G. L. Re, and M. Morana, “Human Activity Recognition Process Using 3-D Posture Data,” *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 586–597, Oct 2015.

[91] I. Theodorakopoulos, D. Kastaniotis, G. Economou, and S. Fotopoulos, “Pose-based human action recognition via sparse representation in dissimilarity space,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 12–23, 2014.

[92] W. Ding, K. Liu, F. Cheng, and J. Zhang, “STFC: Spatio-temporal feature chain for skeleton-based human action recognition,” *Journal of Visual Communication and Image Representation*, vol. 26, pp. 329 – 337, 2015.

[93] R. Slama, H. Wannous, M. Daoudi, and A. Srivastava, “Accurate 3D action recognition using learning on the Grassmann manifold,” *Pattern Recognition*, vol. 48, no. 2, pp. 556 – 567, 2015.

[94] R. Vemulapalli, F. Arrate, and R. Chellappa, “Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 588–595.

[95] R. Anirudh, P. Turaga, J. Su, and A. Srivastava, “Elastic functional coding of human actions: From vector-fields to latent variables,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3147–3155.

[96] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona, “Action recognition from depth maps using deep convolutional neural networks,” *IEEE Transactions on Human-Machine Systems*, pp. 1–12, 2015.

[97] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[98] J. Sung, C. Ponce, B. Selman, and A. Saxena, “Unstructured human activity detection from RGBD images,” in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 842–849.

[99] H. S. Koppula, R. Gupta, and A. Saxena, “Learning Human Activities and Object Affordances from RGB-D Videos,” *International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[100] B. Ni, G. Wang, and P. Moulin, “RGBD-HuDaAct: A color-depth video database for human daily activity recognition,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, pp. 1147–1153.

[101] S. Fothergill, H. M. Mentis, P. Kohli, and S. Nowozin, “Instructing people for training gestural interactive systems,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 2012, pp. 1737–1746.

[102] A. Kurakin, Z. Zhang, and Z. Liu, “A real time system for dynamic hand gesture recognition with a depth sensor,” in *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, Aug 2012, pp. 1975–1979.

[103] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, “Berkeley MHAD: A comprehensive Multimodal Human Action Database,” in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, Jan 2013, pp. 53–60.

[104] V. Bloom, D. Makris, and V. Argyriou, “G3D: A gaming action dataset and real time action recognition evaluation framework,” in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, pp. 7–12.

[105] L. Seidenari, V. Varano, S. Berretti, A. Del Bimbo, and P. Pala, “Recognizing Actions from Depth Cameras as Weakly Aligned Multi-part Bag-of-Poses,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2013, pp. 479–485.

[106] Z. Cheng, L. Qin, Y. Ye, Q. Huang, and Q. Tian, “Human Daily Action Analysis with Multi-view and Color-Depth Data,” in *Computer Vision - ECCV 2012. Workshops and Demonstrations*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7584, pp. 52–61.

[107] C. Wolf, J. Mille, E. Lombardi, O. Celiktutan, M. Jiu, M. Baccouche, E. Dellandréa, C.-E. Bichot, C. Garcia, and B. Sankur, “The LIRIS Human activities dataset and the ICPR 2012 human activities recognition and localization competition,” LIRIS Laboratory, Tech. Rep. RR-LIRIS-2012-004, March 2012.

[108] G. Yu, Z. Liu, and J. Yuan, “Discriminative orderlet mining for real-time recognition of human-object interaction,” in *Computer Vision – ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part V*, D. Cremers, I. Reid, and M.-H. Saito, Hideoand Yang, Eds. Cham: Springer International Publishing, 2015, pp. 50–65.

[109] F. Negin, F. Özdemir, C. Akgül, K. A. Yüksel, and A. Erçil, “A Decision Forest Based Feature Selection Framework for Action Recognition from RGB-Depth Cameras,” in *Image Analysis and Recognition*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7950, pp. 648–657.

[110] C. Chen, R. Jafari, and N. Kehtarnavaz, “UTD-MHAD: a multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor,” in *Image Processing (ICIP), 2015 IEEE International Conference on*, Sept 2015, pp. 168–172.

[111] M. Munaro, G. Ballin, S. Michieletto, and E. Menegatti, “3D flow estimation for human action recognition from colored point clouds,” *Biologically Inspired Cognitive Architectures*, vol. 5, pp. 42–51, 2013.

[112] A. Shimada, K. Kondo, D. Deguchi, G. Morin, and H. Stern, "Kitchen Scene Context Based Gesture Recognition: A Contest in ICPR2012," in *Advances in Depth Image Analysis and Applications: International Workshop, WDIA 2012, Tsukuba, Japan, November 11, 2012, Revised Selected and Invited Papers*, X. Jiang, O. R. P. Bellon, D. Goldgof, and T. Oishi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 168–185.

[113] A. Eweiwi, M. Cheema, C. Bauckhage, and J. Gall, "Efficient pose-based action recognition," in *Computer Vision – ACCV 2014*, ser. Lecture Notes in Computer Science, D. Cremers, I. Reid, H. Saito, and M.-H. Yang, Eds. Springer International Publishing, 2015, vol. 9007, pp. 428–443.

[114] M. Jiang, J. Kong, G. Bebis, and H. Huo, "Informative joints based human action recognition using skeleton contexts," *Signal Processing: Image Communication*, vol. 33, pp. 29 – 40, 2015.

[115] A. A. Chaaraoui, J. R. Padilla-Lepez, P. Climent-Perez, and F. Florez-Revuelta, "Evolutionary joint selection to improve human action recognition with RGB-D devices," *Expert Systems with Applications*, vol. 41, no. 3, pp. 786 – 794, 2014.

[116] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," Ph.D. dissertation, Fairfax, VA, USA, 2004.

[117] A. A. Chaaraoui and F. Florez-Revuelta, "Optimizing human action recognition based on a cooperative coevolutionary algorithm," *Engineering Applications of Artificial Intelligence*, vol. 31, pp. 116–125, 2014.

[118] M. Vrigkas, C. Nikou, and I. A. Kakadiaris, "A review of human activity recognition methods," *Frontiers in Robotics and AI*, vol. 2, p. 28, 2015.

[119] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, and C. Tang, "RGB-D-based action recognition datasets: A survey," *Pattern Recognition*, vol. 60, pp. 86 – 105, 2016.

[120] M. Firman, "RGBD Datasets: Past, Present and Future," *arXiv preprint arXiv:1604.00999*, 2016.

[121] J. R. Padilla-López, A. A. Chaaraoui, and F. Flórez-Revuelta, "A discussion on the validation tests employed to compare human action recognition methods using the MSR Action3D dataset," *CoRR*, vol. abs/1407.7390, 2014.

[122] Microsoft Kinect SDK, https://www.microsoft.com/en-us/download/details.aspx?id=40278, accessed: 2017-02-12.

[123] OpenNI SDK, http://structure.io/openni, accessed: 2017-02-12.

[124] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, "A depth-based joints estimation algorithm for get up and go test using kinect," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2014, pp. 226–227.

[125] E. Cippitelli, S. Gasparrini, S. Spinsante, and E. Gambi, "Kinect as a Tool for Gait Analysis: Validation of a Real-Time Joint Extraction Algorithm Working in Side View," *Sensors*, vol. 15, no. 1, pp. 1417–1434, 2015.

[126] E. Cippitelli, S. Gasparrini, S. Spinsante, E. Gambi, F. Verdini, L. Burattini, F. Di Nardo, and S. Fioretti, "Validation of an optimized algorithm to use Kinect in a non-structured environment for Sit-to-Stand analysis," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug 2015, pp. 5078–5081.

[127] NASA, Staff of Anthropology Research Project, *Anthropometric Source Book Volume II: A Handbook of Anthropometric Data*, ser. Anthropometric Source Book. Johnson Space Center, Houston, TX, USA: NASA, 1978.

[128] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "SHIMMER$^{\text{TM}}$– a wireless sensor platform for noninvasive biomedical research," *Sensors Journal, IEEE*, vol. 10, no. 9, pp. 1527–1534, 2010.

[129] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier, "Fall detection from depth map video sequences," in *Toward Useful Services for Elderly and People with Disabilities: 9th International Conference on Smart Homes and Health Telematics, ICOST 2011, Montreal, Canada, June 20-22, 2011. Proceedings*, B. Abdulrazak, S. Giroux, B. Bouchard, H. Pigot, and M. Mokhtari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 121–128.

[130] G. Mastorakis and D. Makris, "Fall detection system using kinect's infrared sensor," *Journal of Real-Time Image Processing*, vol. 9, no. 4, pp. 635–646, 2014.

[131] V. Bevilacqua, N. Nuzzolese, D. Barone, M. Pantaleo, M. Suma, D. D'Ambruoso, A. Volpe, C. Loconsole, and F. Stroppa, "Fall detection in indoor environment with kinect sensor," in *Innovations in Intelligent Systems and Applications (INISTA) Proceedings, 2014 IEEE International Symposium on*, June 2014, pp. 319–324.

[132] A. T. Nghiem, E. Auvinet, and J. Meunier, "Head detection using kinect camera and its application to fall detection," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, July 2012, pp. 164–169.

[133] M. Kepski and B. Kwolek, "Unobtrusive fall detection at home using kinect sensor," in *Computer Analysis of Images and Patterns: 15th International Conference, CAIP 2013, York, UK, August 27-29, 2013, Proceedings, Part I*, R. Wilson, E. Hancock, A. Bors, and W. Smith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 457–464.

[134] R. Planinc and M. Kampel, "Introducing the use of depth data for fall detection," *Personal and Ubiquitous Computing*, vol. 17, no. 6, pp. 1063–1072, 2013.

[135] C. K. Lee and V. Y. Lee, "Fall detection system based on kinect sensor using novel detection and posture recognition algorithm," in *Inclusive Society: Health and Wellbeing in the Community, and Care at Home: 11th International Conference on Smart Homes and Health Telematics, ICOST 2013, Singapore, June 19-21, 2013. Proceedings*, J. Biswas, H. Kobayashi, L. Wong, B. Abdulrazak, and M. Mokhtari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 238–244.

[136] S.-W. Yang and S.-K. Lin, "Fall detection for multiple pedestrians using depth image processing technique," *Computer Methods and Programs in Biomedicine*, vol. 114, no. 2, pp. 172–182, 2014.

[137] R. Dubey, B. Ni, and P. Moulin, "A depth camera based fall recognition system for the elderly," in *Image Analysis and Recognition: 9th International Conference, ICIAR 2012, Aveiro, Portugal, June 25-27, 2012. Proceedings, Part II*, A. Campilho and M. Kamel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 106–113.

[138] A. Davari, T. Aydin, and T. Erdem, “Automatic fall detection for elderly by using features extracted from skeletal data,” in *Electronics, Computer and Computation (ICECCO), 2013 International Conference on*, Nov 2013, pp. 127–130.

[139] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “An Integrated Approach to Fall Detection and Fall Risk Estimation Based on RGB-Depth and Inertial Sensors,” in *to appear in ACM International Conference Proceedings Series - International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion*, Dec 2016.

[140] E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, J. Wåhslén, I. Orhan, and T. Lindh, “Time synchronization and data fusion for RGB-Depth cameras and inertial sensors in AAL applications,” in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 265–270.

[141] S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante, J. Wåhslén, I. Orhan, and T. Lindh, “Proposal and Experimental Evaluation of Fall Detection Solution Based on Wearable and Depth Data Fusion,” in *ICT Innovations 2015*, ser. Advances in Intelligent Systems and Computing, S. Loshkovska and S. Koceski, Eds. Springer International Publishing, 2016, vol. 399, pp. 99–108.

[142] J. Wåhslén, I. Orhan, and T. Lindh, “Local time synchronization in bluetooth piconets for data fusion using mobile phones,” in *Body Sensor Networks (BSN), 2011 International Conference on*, May 2011, pp. 133–138.

[143] QueryPerformanceCounter, https://msdn.microsoft.com/it-it/library/windows/desktop/ms644904(v=vs.85).aspx, accessed: 2017-02-12.

[144] QueryPerformanceFrequency, https://msdn.microsoft.com/en-us/library/windows/desktop/ms644905(v=vs.85).aspx, accessed: 2017-02-12.

[145] Freescale Semiconductor MMA7260QT datasheet, https://www.nxp.com/files/sensors/doc/data_sheet/MMA7260QT.pdf, accessed: 2017-02-12.

[146] TST Datasets, http://www.tlc.dii.univpm.it/blog/databases4kinect, accessed: 2017-02-12.

[147] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. J “Comparison of low-complexity fall detection algorithms for body attached accelerometers,” *Gait & Posture*, vol. 28, no. 2, pp. 285–291, 2008.

[148] R. Khusainov, D. Azzi, I. E. Achumba, and S. D. Bersch, “Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations,” *Sensors*, vol. 13, no. 10, pp. 12 852–12 902, 2013.

[149] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “A Human Activity Recognition System Using Skeleton Data from RGBD Sensors,” *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

[150] E. Cippitelli, E. Gambi, S. Spinsante, and F. Florez-Revuelta, “Human Action Recognition Based on Temporal Pyramid of Key Poses Using RGB-D Sensors,” in *Advanced Concepts for Intelligent Vision Systems: 17th International Conference, ACIVS 2016, Lecce, Italy, October 24-27, 2016, Proceedings*, J. Blanc-Talon, C. Distante, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer International Publishing, 2016, pp. 510–521.

[151] ——, “Evaluation of a skeleton-based method for human activity recognition on a large-scale RGB-D dataset,” in *Technologies for Active and Assisted Living (TechAAL), IET International Conference on*, Oct 2016.

[152] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[153] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[154] D. Faria, C. Premebida, and U. Nunes, “A probabilistic approach for human everyday activities recognition using body motion from RGB-D images,” in *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on*, 2014, pp. 732–737.

[155] G. I. Parisi, C. Weber, and S. Wermter, “Self-Organizing Neural Integration of Pose-Motion Features for Human Action Recognition,” *Frontiers in Neurorobotics*, vol. 9, no. 3, 2015.

[156] J. Shan and S. Akella, “3D human action segmentation and recognition using pose kinetic energy,” in *2014 IEEE International Workshop on Advanced Robotics and its Social Impacts*, Sept 2014, pp. 69–75.

[157] O. Çeliktutan, C. Wolf, B. Sankur, and E. Lombardi, "Fast Exact Hyper-graph Matching with Dynamic Programming for Spatio-temporal Data," *Journal of Mathematical Imaging and Vision*, pp. 1–21, March 2014.

[158] S. Azary and A. Savakis, "Grassmannian Sparse Representations and Motion Depth Surfaces for 3D Action Recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2013, pp. 492–499.

[159] ——, "3D Action Classification Using Sparse Spatio-temporal Feature Representations," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7432, pp. 166–175.

[160] A. A. Chaaraoui and F. Flórez-Revuelta, "Adaptive human action recognition with an evolving bag of key poses," *IEEE Transactions on Autonomous Mental Development*, vol. 6, no. 2, pp. 139–152, June 2014.

[161] S. C. Akkaladevi and C. Heindl, "Action recognition for human robot interaction in industrial applications," in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, Nov 2015, pp. 94–99.

[162] E. Ghorbel, R. Boutteau, J. Boonaert, X. Savatier, and S. Lecoeuche, "3D real-time human action recognition using a spline interpolation approach," in *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*, Nov 2015, pp. 61–66.

[163] G. Evangelidis, G. Singh, R. Horaud *et al.*, "Skeletal Quads: Human Action Recognition Using Joint Quadruples," in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR 2014)*, 2014.

[164] C. Chen, K. Liu, and N. Kehtarnavaz, "Real-time human action recognition based on depth motion maps," *Journal of Real-Time Image Processing*, pp. 1–9, 2013.

[165] L. Lo Presti, M. La Cascia, S. Sclaroff, and O. Camps, "Hankelet-based dynamical systems modeling for 3D action recognition," *Image and Vision Computing*, vol. 44, pp. 29 – 43, 2015.

[166] L. Tao and R. Vidal, "Moving Poselets: A Discriminative and Interpretable Skeletal Motion Representation for Action Recognition," in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, Dec 2015, pp. 303–311.

[167] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1110–1118.

[168] C. Chen, R. Jafari, and N. Kehtarnava, "Action Recognition from Depth Sequences Using Depth Motion Maps-based Local Binary Patterns," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa Beach, HI, 2015, pp. 1092–1099.

[169] H. Xu, E. Chen, C. Liang, L. Qi, and L. Guan, "Spatio-Temporal Pyramid Model based on depth maps for action recognition," in *Multimedia Signal Processing (MMSP), 2015 IEEE 17th International Workshop on*, Oct 2015, pp. 1–6.

[170] A. Shahroudy, T. T. Ng, Q. Yang, and G. Wang, "Multimodal Multipart Learning for Action Recognition in Depth Videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[171] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[172] E. Ohn-Bar and M. Trivedi, "Joint Angles Similarities and HOG2 for Action Recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2013, pp. 465–470.

[173] J. F. Hu, W. S. Zheng, J. Lai, and J. Zhang, "Jointly learning heterogeneous features for RGB-D activity recognition," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 5344–5352.

[174] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-Temporal LSTM with Trust Gates for 3D Human Action Recognition," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 816–833.

[175] B. Kwolek and M. Kepski, “Human fall detection on embedded platform using depth maps and wireless accelerometer,” *Computer Methods and Programs in Biomedicine*, vol. 117, no. 3, pp. 489–501, 2014.

[176] Kinect Stream Saver Application, https://kinectstreamsaver.codeplex.com/, accessed: 2017-02-12.

[177] S. Gasparrini, E. Cippitelli, S. Spinsante, and E. Gambi, “A depth-based fall detection system using a kinect® sensor,” *Sensors*, vol. 14, no. 2, pp. 2756–2775, 2014.

[178] S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante, and F. Florez-Revuelta, “Performance analysis of self-organising neural networks tracking algorithms for intake monitoring using kinect,” in *Technologies for Active and Assisted Living (TechAAL), IET International Conference on*, Nov 2015, pp. 1–6.

[179] OpenCV - Open Source Computer Vision, http://opencv.org/, accessed: 2017-02-12.