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

Activity Monitoring and Behaviour Analysis Using RGB-Depth Sensors and Wearable Devices for Ambient Assisted Living Applications

Ph.D. Dissertation of:
Samuele Gasparrini

Advisor:
Prof. Ennio Gambi

Curriculum Supervisor:
Prof. Franco Chiaraluce

XIV edition - new series

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

Activity Monitoring and Behaviour Analysis Using RGB-Depth Sensors and Wearable Devices for Ambient Assisted Living Applications

Ph.D. Dissertation of:
Samuele Gasparrini

Advisor:
Prof. Ennio Gambi

Curriculum Supervisor:
Prof. Franco Chiaraluce

XIV edition - new series

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

How do you eat an elephant?
One bite at a time.

Acknowledgments

I would like to express my gratitude to my advisor Prof. Ennio Gambi and his assistant Dr. Susanna Spinsante for their time, constant support and assistance during these years. You gave me the opportunity to grow as researcher and person.

I would also like to thank Automa, the company that has financed my Ph.D. grant.

I also thank Prof. Thomas Lind, Jonas Wåhslén and Ibrahim Orhan from the KTH-Royal Institute of Technology, who assisted me during the STSM in Stockholm.

Many thanks also to Prof. Francisco Flórez-Revuelta for his suggestions and support during the collaboration at the Kingston University. A special thank to the people in the SB1018 laboratory at Penrhyn Road campus, in particular for the cakes during the reading groups!

A thank goes to the colleagues and friends of the "Lunch group" of the Department of Information Engineering at the Polytechnic University of Marche; we have spent enjoyable moments together. A special thank to Enea Cippitelli who helped me during these years and without him most of the results would have not been achieved.

Finally, a thank to my family and my girlfriend Francesca. I cannot thank you enough for the support that you gave me during these years.

Ancona, January 2016

Samuele Gasparrini

Sommario

Nei paesi sviluppati, la percentuale delle persone anziane è in costante crescita. Questa condizione è dovuta ai risultati raggiunti nel capo medico e nel miglioramento della qualità della vita. Il trend è ulteriormente rafforzato dal basso livello di nascite in Europa, che porterà all'incremento del rapporto tra popolazione anziana e attiva dal 27.8% al 50.1% nel periodo 2015-2060.

Con l'avanzare dell'età, le persone sono più soggette a malattie correlate con l'invecchiamento. Esse sono classificabili in tre gruppi: fisiche, sensoriali e mentali. Come diretta conseguenza dell'aumento della popolazione anziana ci sarà quindi una crescita dei costi nel sistema sanitario, che dovrà essere affrontata dalla EU nei prossimi anni.

Una possibile soluzione a queste sfide è l'utilizzo della tecnologia. In particolare, diversi servizi e dispositivi possono essere utilizzati a casa, per rendere la vita delle persone anziane più confortevole e allo stesso tempo indipendente. Questo concetto porta a dei vantaggi diretti, in primo luogo l'ospedalizzazione della persona viene ritardata il più possibile e permette alla stessa di continuare a vivere nella propria abitazione, tenendo conto delle sue abilità, limitazioni e condizioni fisiche. In secondo luogo, la tecnologia può essere utilizzata per supportare l'attività del caregiver, grazie alla possibilità di monitorare lo stato della persona senza la necessità di una visita diretta.

Questo concetto è chiamato Ambient Assisted living (AAL) e copre diverse aree quali ad esempio il supporto alla mobilità, la cura delle persone, la privacy, la sicurezza e le interazioni sociali.

In questa tesi, due differenti famiglie di sensori saranno utilizzate per mostrare le potenzialità della tecnologia nel contesto dell'AAL. In particolare verranno utilizzati due tipi di dispositivi: telecamere RGB-profondità e sensori indossabili. I sistema della prima famiglia, forniscono un'informazione molto utile essendo simile al sistema di visione umano. Sfortunatamente, se la camera è posizionata in una configurazione fissa, la ridotta area monitorata rappresenta una limitazione per questo tipo di sensore.

Per quanto riguarda l'altra famiglia di dispositivi, il problema precedente viene superato, dato che il dispositivo può essere sempre indossato dalla persona. Le limitazioni di questi sistemi si presentano però quando è necessario monitorare specifici comportamenti della persona che possono portare a situazioni di pericolo. Essi sono caratterizzati da fattori complessi che non possono essere identificati solo tramite l'utilizzo di sensori indossabili (accelerometri, giroscopi e barometri).

Negli ultimi anni, l'approccio data-fusion ha attirato l'attenzione della comunità scientifica. Combinando infatti le informazioni fornite dai dispositivi indossabili e dai sistemi video è possibile migliorare le prestazioni dell'applicazione finale. In questo lavoro, vengono trattate le problematiche di sincronizzazione tra i due tipi di sensori descritti in precedenza al fine di sviluppare tre applicazione nel contesto AAL.

La prima applicazione sfrutta una telecamera di profondità per monitorare la distanza tra sensore

e persona al fine di individuare possibili cadute. Un'implementazione alternativa usa l'informazione di profondità sincronizzata con l'accelerazione fornita da un dispositivo indossabile per classificare le attività realizzate dalla persona in due gruppi: Activities of Daily Living (ADL) e cadute.

Al fine di valutare il fattore di rischio caduta negli anziani, la seconda applicazione usa la stessa configurazione descritta in precedenza per misurare i parametri cinematici del corpo durante un test clinico chiamato "Timed Up and Go".

Infine, la terza applicazione monitora i movimenti della persona durante il pasto con l'obbiettivo di valutare se il soggetto sta seguendo una dieta corretta. In particolare, l'informazione di profondità viene sfruttata per riconoscere particolari azioni mentre quella RGB per classificare oggetti di interesse come bicchieri o piatti posizionati sul tavolo.

Abstract

Nowadays, in the developed countries, the percentage of the elderly is constantly growing. This situation is a consequence of improvements in people's quality life and developments in the medical field. At the same time, the low fertility condition all over Europe will increase this trend, with forecasts that state a demographic old-age dependency ratio estimated to raise from 27.8% to 50.1% in the period 2015-2060. Because of ageing, people have higher probability to be affected by age-related diseases classified in three main groups: physical, perceptual and mental. Therefore, the direct consequence is a growing of healthcare system costs and a not negligible financial sustainability issue which the EU will have to face in the next years.

One possible solution to tackle this challenge is exploiting the advantages provided by the technology. In particular, different services and devices can be used at home, in order to have a living space more suitable as well as comfortable for the independent life of the elderly and also to be adaptable to the users' needs.

There are many positive advantages behind this living concept. First of all, the hospitalization of the elderly is delayed as long as possible and allows them to independently live at their home, considering their abilities, impairments and chronic conditions. Secondly, the technology could be a support to the caregiver's activities, thanks to the possibility to monitor and assess the health status of the person without a need of a doctor's visit.

This paradigm is called Ambient Assisted Living (AAL) and concerns different areas, such as mobility support, health and care, privacy and security, social environment and communication.

In this thesis, two different types of sensors will be used to show the potentialities of the technology in the AAL. Low-cost RGB-Depth cameras and wearable devices will be studied to design affordable solutions. The first sensor family is very powerful as it provides information quite similar to the human vision which is used to understand the surrounding environment. Unfortunately, if the camera is in a fixed position, the finite covered area represents a limit for this sensor. On the other hand, the wearable devices can overcome the problem because the user always wears them. They integrate heterogeneous sensors like accelerometer, gyroscope and barometer to measure the most important parameters of the user's health status. However, when the behaviours of the patient must be monitored to infer possible risk situations, some limitations arise. They are due to the complex factors that characterize these movements which cannot be measured using wearable sensors only.

In the last years, the so-called data fusion approach has drawn the attention of the research community. Indeed, the combination of wearable and camera devices can improve the performance of the final application. In this work, the synchronization issues between RGB-Depth cameras and wearable sensors are addressed and three AAL applications are designed.

The first one is a fall detection system that uses the distance information (depth) between the target and the camera to monitor more people inside the covered area. The application will trigger an alarm when recognizes a fall. An alternative implementation of the same solution synchronizes the information provided by a depth camera and a wearable device to classify the activities performed by the user in two groups: Activities of Daily Living (ADL) and fall.

In order to assess the fall risk in the elderly, the second proposed application uses the previous sensor configuration to measure kinematic parameters of the body during a specific assessment test called "Timed Up and Go". Finally, the third application monitor's the user's movements during an intake activity. Especially, the drinking gesture can be recognized by the system using the depth information to track the hand movements and the RGB stream of the camera to classify important objects like a glass placed on a table.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AAL	Ambient Assisted Living
ADL	Activities of Daily Living
AmI	Ambient Intelligence
API	Application Program Interface
BMU	Best Matching Unit
BOF	Beginning Of File
BT	Bluetooth
CCD	Charge-Coupled Device
CCS	Camera Coordinate System
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
CWIM	Continuous Wave Intensity Modulation
DCT	Discrete Cosine Transform
EOF	End Of File
EU	European Union
FOV	Field Of View
FRS	Frame Reference System
GNG	Growing Neural Gas
GUI	Graphic User Interface
IDE	Integrated Development Environment
IR	Infrared

LIDAR	Light Detection and Ranging
NG	Neural Gas
Pc	Personal Computer
PC	Point Cloud
SAR	Socially Assistive Robot
SBI	Suppression of Background Intensity
SDK	Software Development Kit
SHIMMER	Sensing Health with Intelligence, Modularity, Mobility and Experimental Reusability
SOM	Self-Organizing Map
SOM_Ex	Extended Self-Organizing Map
TOF	Time Of Flight
TUG	Timed Up and Go
WHO	World Health Organization

Chapter 1.

Introduction

1.1. Contest

The word "elderly" has generic meaning and the related age varies between the different countries. For example, in Europe a citizen is considered old if he is over 60, whereas in Japan the limit grows to 75. As defined by the World Health Organization (WHO) "there is no United Nations (UN) standard numerical criterion, but the UN agreed cut-off is 60+ years to refer to the older population" [1]. Laslett [2] introduced a specific classification for the elderly: from 65 to 80 years old they belong to the third age, while getting older than 80 they enter in the fourth age. In the first group people are usually enjoying their retirement in a healthy self-reliant and active health status. In the second group, the elderly have less independence and their physical condition starts to have biological and psychological complications. They reduce physical activities, have less appetite with a consequent weight loss and their mental capacity is day by day affected by chronic diseases.

The Department of Economic and Social Affairs (DESA) of UN states that in 2015 there are approximately 900 million people aged 60 or over, with a global share of 12 per cent of the population [3]. Most of the developed countries' populations are ageing and, although the demographic profile presents some differences from continent to continent, there are some significant similarities between these countries. According to this theory, in 2060 the number of people that belong to the third age will increase from 17.4% to 29.5% in Europe, from 13.2% to 21.9% in the USA, from 22.7% to 35.1% in Japan and from 8.2% to 29.5% in China [4]. This phenomenon will lead to impressive numbers: in 2060 there will be 2 billion people over 60 years old and only 440 million on China.

Even thought the European population is expected to increase about 5% from 507 million in 2013 to 523 million in 2060 [5], currently the old continent has the highest percentage of people aged 60 or over in its population [3] with a growing of approximately two million per year. In details, people over 50 will raise by 35%, whereas citizens aged from 65 to 80 years old will grow by nearly 40% between 2010 and 2030 [5] and finally the number of the elderly over 85 will triple by 2050.

On the other hand, in the old continent, there is also a low level of fertility issue. The replacement rate was 1.6 children per women in 2010-2015 and will grow to 1.8 by 2050 [3], but it is always below the natural value of 2.1 to achieve a total replacement of the population.

In the European Union (EU), the low fertility condition and the ageing of the population lead to an important change in terms of age dependency ratio. The most difficult condition is expected during the period 2015-35 when the so called "baby-boom" generation will retire. In particular, in the period 2015-2060, the share of young people (age 0-19) will remain close to 20% in the EU and the percentage of citizens aged from 20-64 will reduce from 61% to 51%. People in the third age will grow in share from 14% to 20%, whilst the fourth age group will rise from 5% to 10%. The old age dependency ratio, that is the ratio between the population aged 65 over the people aged 15-64, is estimated to increase from 27.8% to 50.1% [5]. Consequently, if today in the EU for a single person in retirement there are four "active" people belonging to the working-age population, by 2025 this ratio will start to drop to 3 and in 2060 it will be even 2 [5].

Due to the physical and mental disorders, people in third and fourth ages have a higher probability to be affected by different types of age-related diseases than the younger population. Indeed, only 2.7% of the 25-34 population suffers of severely hampered compared to the 13.9% of people aged 55-64 and 39.1% of people over 85 [6]. These diseases are classified as physical, when related to motor disabilities that limit balance and mobility, or chronic such as heart disease, hypertension, chronic respiratory diseases, diabetes and heart failure. Perceptual diseases affect the senses such as sight and hearing. Finally, among the mental ones, the most important and spread are the Alzheimer and Parkinson's diseases. For the first one, today people who suffer it are 26.6 million worldwide and they are expected to be 80 million in 2050 [7], whereas the Parkinson's disease, according to WHO, affects about 3% of the population over 65 years old [8].

The aging population trend, the corresponding healthcare system costs and the pension system will increase in the future and the EU will spend almost 20% (4.1 percentage points of GDP) in the period 2010 - 2060 [5]. Therefore, there is a financial sustainability challenge facing the EU to maintain the current assistance level and in general the elderly quality life.

Nowadays, if possible, most of the elderly prefer to live in their own houses with their spouse or relatives. Care of them is usually the responsibility of the healthcare system, but due to the shortage of professionals of approximately close to 2 million workers by the year 2020 [5], the family members take the role of informal caregivers. It could be a very grateful and gratifying activity with the possibility of a personal grow. However, this also generates some negative consequences for the relatives of the recipient of care, such as emotional distress and frustration when they are not able to solve emergency situations, provoking a burnout as ultimate result. There are some drawbacks also in their job because of workplace disruptions, absenteeism and reduced work time to take care of the older adults, as well as a consequent salary reduction [9]. Sometimes a full-time job does not allow taking care of the parents and the elderly will not receive the adequate level of care. Therefore, the informal caregivers are often busy between their need and their responsibilities towards the elderly.

In the last years, there was an effort to solve these problems providing different services and devices to the elderly through dedicated technologies installed in the so-called "intelligent" home. This vision relies on the potentialities of pervasive information and communication technology (ICT) to make the house environment adaptable to the users' needs. The Ambient Intelligence

(AmI) is used to denote this condition, where different technological and scientific contributions such as human-machine interactions (HMI), wearable/ambient sensors and behaviours analysis solutions are combined together to have a digital environment that changes along with the characteristics of the people who live inside it [10].

One of the most significant application of this paradigm is the Ambient Assisted living (AAL). Its aim is strongly related to the AmI, but it is finalized to improve the quality life of elderly people through a personal healthcare [11]. Considering their abilities, impairments and chronic conditions, the AAL allows the elderly to live independently in their home, for as long as possible in order to delay the hospitalization at the last part of their life. In this way, there are two advantages: the health care expenditure can be kept below the budget limits exploiting cost-effective solutions that use the ICT and, at the same time, the elderly can continue living in their preferred environment with their loved.

A complete definition of the AAL is given by Kung et al. [12]: "intelligent systems that will assist elderly individuals for a better, healthier and safer life in the preferred living environment and covers concepts, products and services that interlink and improve new technologies and the social environment". Therefore, the system will address specific tasks such as:

- health and care: support elderly users during their Activities of Daily Living (ADL), such as dressing, grooming, bathing and eating [13]. At the same time, monitor chronic diseases that lead to different behaviours from the ADL or in general, use specific wearable sensors to control the health status of the patient (blood pressure, blood sugar and heart rate) [14]. It can also implement reminder functions to intake pills or rehabilitation session through serious game [15];

- privacy and security: promptly identify emergency situations directly related to the health status of the elderly, for example after a fall. The risks inside the living space are other examples of abnormal situations, i.e. stove-top left on or in general, appliances that are incorrectly used. In addition, the relatives or the caregivers will be alerted if the abnormal situation persists in order to solve the problem in the case that an automatic solution is not possible [16]. One of the most important characteristic that an AAL must comply is the "invisibility". It means that the system will intervene only if it is really necessary, so that the user preserves his self-sufficiency;

- social environment and communication: monitor and establish interaction with other people inside/outside the home to prevent loneliness and improve social skills. Thanks to the ICT solution is possible to promote communication between the patient and the doctor, keep in touch the elderly and the family members who live away from them. The socially assistive robot (SAR) is another example of social interaction, in this case between the elderly and an interactive robot that reacts to human stimulus to address specific needs [17];

- mobility support: an AAL application can be used also to help the elderly outside their houses through orientation and navigation instructions to the final destination passing through the safest itinerary.

These are just some of the possible applications of the technology to the home environment that lead to more comfortable and suitable space for the independent life of the elderly.

To achieve this result the AAL paradigm cannot be seen as a single closed element, but it must realize a relationship network among heterogeneous partners. They belong to different areas such as medical, social, technological and business and they have high-level specifications which have never been combined in a single solution. These actors can be classified in four groups:

- primary: seniors or impaired citizens who are using the AAL applications;
- secondary: people close to the primary users (family members and relatives of the recipient of care, private or public caregivers and neighbours). They are the subjects who gain advantages from the direct use of AAL products [18];
- tertiary: stakeholders or in general the companies that provide services to the final user (medical professionals, small and medium-sized enterprises (SMEs), insurance companies, housing associations, service organizers, security companies);
- quaternary: private or public organisations and institutions that do not use directly the AAL solutions, but push the networking among the previous actors with grants and project funding (consortia, policy makers, non-governmental organizations, academia).

In order to promote the interaction between some of these subjects, the EU has been dedicated a considerable number of project calls in the sixth and seventh framework programmes, respectively in the periods 2002-2006 and 2007-2013. Today, their successor is Horizon 2020, the new framework programme on innovation and research with 80 billion budget from 2014 to 2020. It confirms the project vision of the EU in supporting the applications designed for the independent living of its citizens. In particular, the Work Programme 2016-2017 for Societal Challenge 1 related to Health Demographic Change and Wellbeing will try to work in this direction. The Active and Assisted Living Joint Programme (AAL JP) and its successor AAL JP2 are an applied research funding programmes focused on the AAL. Their aim is to design and develop innovative products that can be used to improve the wellbeing of the elderly. Especially, in the period 2008-2013 the AAL JP proposed six calls that produced around 130 projects funded with a total € 317.5 million by the public funding commitment [19]. An introduction of successful AAL projects is described in [20], of course this in not a complete review, but gives some information of the impact that this collaboration has in the elderly population of the EU.

These investments have the headline goal to promote a market growth of ICT technologies in the AAL area increasing job opportunities. This chance, called “silver economy” [21], refers to new industry sectors that provide specific products like smart home automation, remote health monitoring system and companion robots specifically tailored for the elderly. It is also pushed by the spending capacity of the Europeans over 65 that today is estimated over € 3K billion [22].

1.2. Background

When an AAL application must be implemented, among all context-aware systems (passive infrared (PIR) sensors, cameras, microphones, pressure sensors, floor sensors) the RGB camera is the sensor family that provides the richest semantic information to the algorithms designed to monitor the elderly during their daily life at home. However, one of the main limitation of these devices relies on the dependence of the raw data to the environment illumination where the sensor is placed. As a consequence, this often leads to a higher complexity in the algorithm that uses the RGB source. In addition, there is also a privacy issue if the raw data captured from specific rooms, like bathroom and bedroom, is not locally used, but streamed on internet from the private space to the remote processing centre. A possible solution is to encrypt the communication or partially modify the information through a protection method (obscuring or distorting) that directly works on the raw data [23]. In the recent years, the depth cameras have received interest as complementary alternative to the previous sensor. Instead of colour information, the depth frame stores in each pixel the distance between the camera and a specific point of the scene. This sensor can work well in dark rooms with poor light conditions and it is less intrusive than the RGB camera because, through the distance, the identity of the person is respected. Therefore, the acceptability and the privacy issues of this type of technology for AAL applications can be significantly increased [24]. However, as for the RGB sensors, some limits are still present. The information is limited by the field of view (FOV) of the camera, as well as if the monitored area is the entire house, in every room must be installed at least a camera. Another limitation of RGB/Depth technologies is the occlusion problem, for example a piece of furniture could partially cover the human shape.

An alternative to the video-based monitoring systems are the wearable devices which could be worn under or over the clothes. They host different sensors (accelerometer, gyroscope and barometer) in the same space and the measured data could also be directly elaborated inside the device system. When the computational costs of the algorithm affect too much the battery drain of the system, another choice is sending the data to an external unit using a dedicated wireless transceiver that is compliant with a specific wireless protocol (Bluetooth, ZigBee, Direct Wi-Fi, proprietary). The receiver is usually connected to the power line and this allows implementing more sophisticated analyses, not limited by a power constraint. In addition, in order to improve the performance, the receiver can collect samples from more than one wearable sensor worn by a person and process them using the same algorithm.

Compared with the camera-based system, the limited covered area is now overcome by the portability and therefore the sensor continues its work also outside the house, but at the same time, it is one of the main drawbacks of this technology. Taking into account that the device will be used for AAL applications, there could be an acceptability issue. Indeed, if the wearable device is cumbersome this can be discriminatory and the community could see the elderly as frail and vulnerable. Consequently, they could refuse to wear the system. Therefore, the device must be comfortable and as discreet as possible.

In the last years, the data-fusion approach has drawn the attention of the research community. The idea is to simultaneously use the data provided by heterogeneous sensors to improve the al-

gorithm performance. For example, when the application uses a vision-based sensor located on the wall together with an acceleration data provided by a wearable device, and in case that a piece of furniture occludes the person, the system can continue tracking the patient thanks to the information from the accelerometer. Of course, there could be synchronization issues that must be considered. They are due to the different sampling time on the devices and possible delays related to the wireless communication protocol used by the wearable solution.

A possible application of sensors previously described is for the automatic fall detection. As stated by the WHO, "A fall is an event which results in a person coming to rest inadvertently on the ground or other lower level". Considering that one every three adults (28-35%) aged 65 or over falls each year and this percentage is higher (32-42%) for people over 70 [25, 26], fall is the most widespread and frequent domestic accident that the elderly undergo at their own houses. The causes of a fall are related to intrinsic (aging, Parkinson's disease, neurodegenerative or balance disorders, vision, muscle impairments) and extrinsic factors (slippery floors, loose carpets, effects of medications, poor lighting, obstacles on stairways, clutter) [27]. Some precautions to reduce the probability of fall can be taken on the extrinsic factors while the intrinsic ones are more difficult to overcome.

Direct consequences of a fall are lacerations, tissue injuries, joint bone fractures, dislocations and head trauma [28]. Whereas long-term effects of a fall can lead to physical disability, mental shock like anxiety, depression and fear to fall again, reduction in quality of life, severe dependencies and, in some cases, the death. It has been proven that the severity of these consequences is also related to the response time of the first assistance. Indeed, when an elderly person remains on the floor for a long period of time (long-lie) due to his impossibility to get-up, the consequences could be hypothermia, dehydration, bronchopneumonia and pressure sores [29] and even a high probability to die within 6 months [30]. As confirmed by statistical offices of the EU28 and the European Economic Area (EEA), approximately 100 K people die per year due to serious consequences of a fall [31]. Therefore, it is strongly recommended a system to monitor elderly activities and trigger an alarm if a risk situation is recognized. At the same time, this type of service brings noticeable advantages not only to the final user, but also to health care system. For example, in terms of saving time, thanks to this support, nurses do not need to monitor the elderly for most of their time, but can assist them only during the emergencies.

The wearable devices were the first solution adopted to deal the problem. Different configurations are possible, for example Bourke et al. [30] placed tri-axial accelerometers in the thigh and the trunk of 10 volunteers. Whereas Kangas et al. [32] tested their algorithms with the data provided by the sensors set to the waist by an elastic belt, close to the body's centre of gravity. However, as stated by Pannurat et al. [28], other configurations are possible (shoulder, head, wrist, armpit, foot, neck), but the most common position is the waist whereas a good trade-off between performance and comfort is the chest. Of course, more than one sensor in different position can be worn at a time [33], but if on one hand it will improve the performance of the system, on the other hand there is a reduction of the comfort.

The algorithms proposed in literature use the raw data provided by the sensors to correctly

classify the activities in two main groups: ADL and falls. The most important activities belonging to the first group are: walking, sitting, grasping an object from the floor and lying on the bed or sofa, whilst different falls are possible according to the fall direction: front, back, side (simple fall) and end up sitting (complex fall).

However, the automatic classification is not always possible and, when this happens, the system throws a "false alarm". Indeed, this is not a trivial task because some ADL have similar features to falls. For example, in [32] most of the errors are due to the "lying down" movements because their acceleration peaks have similar trends in a fall. The features must account discriminative information such as velocity, body orientation and impact. Starting from the acceleration, the most used features are mean, standard deviation, acceleration magnitude and tilt angle between the sensor and the ground plane (or the gravity axis) [28]. The mean and the standard deviation are used to separate static and dynamic activities, whereas the magnitude highlights abnormal situations. Finally, the angle allows calculating the orientation of the body to understand if a person is lying on the floor, for instance after a fall.

In literature, the algorithms that exploit this information, to classify the activities between falls or ADL, are divided in two main groups: threshold-based (or rule-based) and machine learning approach. In the first one, a finite-state machine compares for each block a feature with the corresponding threshold and a fall is identified if all the conditions are true. For example, Kangas et al. [34] design three algorithms with different levels of complexity using three accelerometers placed at the wrist, the waist and the head. The checks implemented by the most complex algorithms use the information of pre-impact acceleration, velocity, impact peak and orientation of the body. A detailed review of the fall detection algorithms that use the acceleration information can be found in [28, 35].

The machine learning solutions use more sophisticated approaches. Different features (maximum, minimum, average, variance, FFT samples, autocorrelation sequences) are calculated starting from the raw data provided by the sensors and recorded during falls and ADL. The calculated values are respectively labelled as ADL or fall and equally distributed in two groups for testing and training operations. The training set will supply a machine learning algorithm and its performances are evaluated by the testing group. Özdemir et al. [33] use samples from six devices placed in different positions of the body to calculate, for each test, the previous features around the acceleration peaks only. Then, to reduce the high number of features, a Principal Component Analysis (PCA) is used to find the first thirty most useful components. Different machine learning solutions (Bayesian Decision Making, Least Squares method, Support Vector Machine (SVM), k-Nearest Neighbour classifier, Dynamic Time Warping and Artificial Neural Networks) are trained and the performances are evaluated by the test dataset. Similar approaches are described in specific section in [28].

Alternative fall detection solutions use single RGB camera [36, 37]. The person's blob is extracted from the RGB frame and several features are then calculated studying the silhouette pose to detect a possible fall. For example, matching an ellipse on the blob and calculating the width/height ratio or, when the head is detected, monitoring its position and velocity.

To face the problem of a monitored area limited by the FOV of the camera, more than one sensors are installed in different rooms and can be used to independently track the person. For

example, Cucchiara et al. [38] installed a single camera per room to monitor human silhouette after a background suppression. In addition, to deal with occlusions, they use a sophisticated tracking algorithm based on probabilistic appearance models. On the other hand, rather than use a single camera per room, multiple sensors that frame the same scene are useful to reconstruct the 3D information of the monitored area. However, the calibration process is a time consuming operation that burdens on overall complexity of the algorithm. Compared to a single RGB camera approach, now the significant advantage is that 3D features can be exploited.

The depth sensor can directly provide the distance information and therefore different configurations are possible. Leone et al. [39] set the sensor on the wall and, after an automatic calibration procedure, find the ground plane. Foreground elements are detected with a Bayesian segmentation on the depth frame and only the blobs with specific ratios are classified as person. The tracking of the blob is implemented in the 3D space and the distance between the centroid of the person's cluster and the floor plane is monitored. A fall is identified when the distance is below a specific threshold and the person stays in that position for at least 4 s. Stone et al. [40] deploy a similar configuration as well as the technique for foreground blob selection. The fall detection approach is different and based on a two-stage technique where 3D-object's features are used to get a confidence parameter of the fall. It is worth noting the dimension of the dataset used by the authors. It is one-year records on 13 different apartments, but unfortunately it has not been published. Due to the specific sensor configuration, it is possible that the person is occluded by the furniture during the final part of the fall, this problem is dealt in [41, 42]. In the first work, after the blob identification, if suddenly the person disappears due to a fall behind an object, the algorithm evaluates the body velocity. This feature is calculated as the displacement of the body's centroid over one second. The system will trigger an alarm if the velocity, calculated in the time interval before the occlusion, exceeds a threshold. Zhang et al. [42] propose specific datasets, with and without occlusions, to compare the different fall detection algorithms suggested in literature that work with depth data. In the occlusion situations, they state that the solution proposed in [41] achieves better performances.

Another strategy to reduce the occlusion problem is to set the depth camera on the ceiling. Kepski et al. [43] proposed this set-up with a depth camera mounted at the height of about 2.5 m from the floor. They assume that the biggest blob in foreground is the person to monitor and, to classify a lying pose situation, they use different features calculated from the binary image. In addition, when the fall event is recognised, the algorithm controls the ratio between the blob's height and the same parameter calculated 1 s before. In this way, the system takes into account fast changes of the head-floor distance during a fall event. The same camera configuration is proposed in [44], but differently from the previous solution, now the system is able to track more than one person inside the monitored area. The foreground blobs are identified with a Gaussian Mixture Model background subtraction algorithm and, for each blob, three features (person's height, head size, head-shoulder distance) are calculated and used to classify the person from the objects. The person's centroid is tracked in all next frames and when two or more people are so close to merge in the same blob, specific segmentation algorithms are used to separate the silhouettes. The fall is detected when the depth value of the head is close to the floor level.

When a person is in front of the camera, specific devices like the Microsoft Kinect in pairs with dedicated software tools [45, 46] directly recognize the subject. Consequently, in this sensor configuration the pre-processing techniques previously described are not necessary and the tools provide useful information to be exploited for fall detection applications. Mastorakis et al. [47] use the 3D bounding box, which encloses the volume of the person, to devise a system that recognizes a risk situation evaluating the velocity whereby the box changes its width, height and depth during a fall. The skeleton, i.e. human structures modelled with the coordinates of the body's joints, is another information in output from the tools. Planinc et al. [48] calculate the body's axis from the 3D coordinates of the head, shoulder centre, spine, hip and knees. The system will trigger an alarm when the angle between the axis and the floor plane is close to 0° and the coordinate of spine joint is near the floor.

The data fusion approach, if compared with each separate solution, is implemented to improve the performance of the final system and reduce the percentage of false alarm. As stated before, if the acceleration data is the only measurement provided by the wearable devices, it is difficult to separate real falls from fall-like activities because some ADL (lying on the bed or fast sitting down) have same acceleration trends. Diraco et al. [49] use a fuzzy rule-based logic to aggregate the information provided by a wearable accelerometer and a depth camera. In particular, exploiting the ZigBee communication, the inertial sensor informs the coordinator system if it is worn by the user and the probability of fall, through a confidence value between 0 and 1. Whereas, the camera provides information about the number of people in the monitored scene, body posture and falls detected. A similar set-up system is proposed in [50], the acceleration peak is used to indicate a potential fall, thus five features, four from the depth frame and one from the corresponding 3D space, are given in input to a k-NN classifier to recognize the lying pose.

To understand the risk factors related to a decrease of physical abilities that lead to falls in aged people, a proactive approach analyses the human gait and posture parameters. Indeed, instead of identifying the risk situation, the alternative could be prevent it. To this end, today different tests (early diagnosis), not only related to a fall experience [51], but also other adverse health-related events, can be exploited to assess these factors. When the risk of falling is identified, adequate treatments can be taken, for example promoting balance training or physical activities. The "Timed Up and Go" (TUG) test is one of the most used fall risk assessment tool and, in addition, it could also be exploited to evaluate the capability of Parkinsonian or stroke patients for rehabilitation purposes [52, 53]. The TUG test is a diagnostic tool used by the hospital staff to assess the functional mobility of a patient while he is performing specific motor exercises. The test starts with a senior sat on a chair and when the clinician gives a signal, he stands up, walks straight for three metres, turns around, walk back and sits again. During the test, the staff use a stopwatch to take note of the time interval required by the patient to complete these movements. The result is a score between 1 (low risk) to 5 (high risk) that is strongly related to the experience of the medical staff, therefore subjective. The technology can be used to measure further kinematic parameters and thus complete the assessment of the patient. To this end, Qi An et al. [54] place accelerometers and gyroscopes on back, middle of the lower leg and thigh to measure the angles of ankle, knee and hip. As well as the fall detection solutions, these sensors could be uncomfortable

and could limit the movements during the test. Furthermore, taking into consideration that it is impossible to infer all the kinematic information from the measurements of a single device, an array of sensors must be correctly worn by the subject. Therefore, the help of a trained medical staff is indispensable. Consequently, the test will not perform in a domestic environment but it has to be carried out in a specific gait analysis laboratory.

A first alternative solution to wearable sensors is an infrared (IR) marker-based system (Vicon, Qualisys, Optitrack) and it allows studying the spatial and temporal gait parameters during the TUG test. It tracks IR markers placed on specific points over patient's clothes or skin using a set of calibrated IR cameras, opportunely positioned in the room. The system is able to find the markers in the IR frames and it calculates the corresponding 3D coordinates. The repetition of this process for all the available frames produces the trajectories of the movements performed by the monitored person. These optical systems have a high accuracy thanks to the remarkable sampling rate (from 100 to 400 Hz) and can precisely calculate the gait parameters [55]. Unfortunately, as for wearable sensors, the patient must wear foreign objects (IR markers) and a specific environment together with expensive equipment are required.

Another solution is suggested by Wang et al. [56]. They proposed a marker-less system based on two synchronized RGB web cameras. One camera is placed in front of the patient, while the second is in the orthogonal direction and both the devices frame the subject at 5 fps. The aim is to calculate the time to complete the turn phase and the number of steps performed during that interval. Although the system is less invasive than the two previous solutions and the authors claim results close to the clinical rating, the performances are not comparable with an IR marker-based system.

A trade-off between comfort and performance is achieved by depth cameras. As stated before, the skeleton engine provided by the Microsoft Kinect is a useful data to assess the gait parameters during a TUG test. In particular, Galna et al. [57] measured the tracking performance of the skeleton in comparison with a gold standard marker-based analysis system on specific movements like multi-directional reaching, standing and walking on the spot. They state that the accuracy of Kinect skeleton is adequate to track relevant movements from a clinical point of view, but the same assertion is not true for small movements such as hand clasping. Lohmann et al. [58] calibrate two Kinect sensors to cover the whole area requested by the TUG test. The analysis of the two synchronized skeleton streams allows identifying nine events in the test. A subset of these instants (start moving, start walking, start turning, end turning, end moving) is compared with the ground truth obtained from the manual analysis of the videos and the results show a difference lower than a second. Reducing the monitored area to 2 m, Kargar et al. [59] use a single Kinect device in front of a person to calculate both the gait parameters (number of steps, step duration, turning duration) and the anatomical parameters (knee angles, leg angle, distance between elbows) of 12 elderly patients. After a processing on the anatomical parameters, all the features are used in input to a linear SVM classifier to distinguish the patients in two groups: high and low risk of falling.

The behaviour analysis related to food intake is another research field where the technology can lead to an important contribution. Indeed, in the last years, through the Seventh Framework

Programme, the EU has promoted this research [60]. Today it continues with new call proposals in the Horizon 2020 framework [61], tailored for elderly people (SFS-16-2015: "Tackling malnutrition in the elderly").

In the light of this, different elements that contribute to the healthy status of the body can be monitored. One of the most important is the water assimilation. Indeed, different studies have found out the minimum amount of water that a person should drink during a day [62]. If not respected, the risk of stroke, kidney failure, fatal coronary heart disease or even colon and breast cancer may arise [63]. For the elderly this problem is particularly important due to the reduction of their thirst sensation [64].

Malnutrition is another problem that affects a correct diet. It is defined as a state in which a deficiency, excess or imbalance of energy, protein and other nutrients cause adverse effects on body form and clinical outcome [65]. Obesity is one of the most important consequence of malnutrition. In 2007 obese people in the world were 300 million [66] and after eight years the number of these people has almost doubled [67]. This problem leads to an increase of the risk factors associated to many health complications, such as digestive disorders, diabetes type 2, depression, cardiovascular diseases and some types of cancer [68]. The "anorexia of aging" [69] is another demonstration of malnutrition, opposite to overweight. The appetite declines with the age increase and, as a consequence, the food consumption declines of approximately 30% between 20 and 80 years old [70]. The age related decrease in food intake is a response to the decline in energy expenditure with age, which is in its turn related to reduced muscle function and decreased bone mass. However, in many older people, the decrease in energy intake is greater than the decrease in energy expenditure and the result is a loss of body weight. This physiological age-related reduction in appetite and energy intake has been termed the "anorexia of aging".

When the technology was not fully mature, the classic solutions to assess the intake behaviours of the patient relied on self-report filled by the same person. The drawback of this approach is in the low quality of the gathered information which is mostly related to the record skills of the patient [71, 72]. Even in monitored structures such as hospital and nursing home, this problem is not completely overcome. The medical staff uses specific balance charts to manually note the patient's feeding status. Unfortunately, also in this case, the result is not objective and, due to significant pressures on their work, some oversights are still possible [73].

Today, in literature, the diet annotation problem has been partially solved with different approaches. The RFID technology is used in [74] to identify how the volunteer interacts with different objects placed on the table during a meal. The chewing sounds are other distinctive information that can be monitored. Especially, some microphones are placed ear canal of the patient during an intake action [75]. Electromyographic (EMG) signals are the possible alternative used by Woda et al. [76] to study mastication, whereas similar results are obtained using microphones applied to the outer ear canal of the subject to recognize chewing sounds [77].

The embedded sensors have been exploited also in this topic to classify the specific action sequences performed by a person during intake activities. In details, Dong et al. [78] used accelerometer and gyroscope applied to the user's wrist, as well as in [79] where, in addition, a notification is sent to the user's smartphone if his intake rate is higher than a typical value. Thomaz et al.

[80] put an off-the-shelf smartwatch device in the dominant hand to record the intake activities exploiting the integrated 3-axis accelerometer. Five features are calculated for each axis using a sliding window of 6 s. Then, a Random Forest machine learning algorithm is trained to classify both eating and non-eating activities.

Another source of information to monitor the intake activities is provided by the vision-based devices. For example, starting from a photo of a meal, the system is able to classify the different foods in the image [81, 82] and provides the calorie consumption and nutrient composition of the dish [83]. Dehais et al. [84] exploited the stereoscopic principle to analyse two consecutive photos of the plate and calculate the food volume. Unfortunately, in these applications, the user must always use a device to take a photo of the meal and, for most of the elderly, this operation could be a problem. A system to automatically identify the best representative image of the meal is proposed in [85]. The RGB camera is placed on the ceiling and it is less invasive if compared with the previous work. Iosifidis et al. [86], during the lunch, use an RGB camera in front of four people to classify the intake activities. A skin colour segmentation algorithm is applied to the RGB stream in order to obtain both the head and hands' blobs. A sequence of processed frames represents a feature to classify eating and drinking activities. In order to correctly select the person's blob, the volunteers must wear long sleeves. As for the previous AAL applications the depth camera can overcome these limitations. For instance, placing the camera in front of the volunteer, the skeleton, calculated from the depth stream provided by the Kinect sensor, is used in [87, 88, 89, 90] to detect when the person is drinking or eating. For the tests described in [87], the person sits near the table and a Kinect sensor is placed in the opposite side. Three people participate to the tests eating a soup and a main course. The authors use skeleton joints of the head-hands to identify when the volunteer moves cutlery to the mouth. They manually labelled also the position of the dish and the glass to understand when one of these objects is used. A threshold-based algorithm classifies both the drinking and eating movements monitoring the distance between joints and objects. Tham et al. [88] focus their work in the drinking recognition. Starting from the skeleton, the depth area around the dominant hand is selected and normalized. The intervals of the depth stream that frame the drinking action are manually selected and the similarities with a reference sequence are calculated by the Dynamic Time Warping algorithm. The same Kinect configuration is used in [89], but now the volunteer also wears two accelerometers on the wrists during drinking/eating activities. The monitored parameters are displacements of skeleton joints, wrists/arms angles and accelerometer magnitude. Burgess et al. [90] use a Naive Bayes classifier to infer fluid intake actions. The performance of the system can decrease if Kinect is not in front of the person or when some movements are occluded. A more sophisticated solution is proposed by Lei et al. [91], the camera is installed on the ceiling and the algorithm uses a Gradient Kernel descriptor and a linear SVM classifier to find and track the hands.

1.3. Thesis Outlines

In this thesis, the three research topics previously described will be analysed through the study and design of AAL applications that use RGB-Depth cameras and wearable sensors. In particular,

the rest of the work is organized as follows:

Chapter 2 introduces the RGB-Depth cameras and the wearable devices. It starts with the description of the Pinhole camera model and moves to the analysis of the different technologies behind the distance measurement, highlighting their strengths and weaknesses. Then, two sections of the chapter are dedicated to Kinect V1 and V2, where their most important features are compared. Concerning the wearable devices, the SHIMMER sensor is introduced. Its characteristics are presented to explain how to handle the raw acceleration data from the embedded sensor to calculate the device orientation and the magnitude information. Finally, two software tools are presented. They are used to collect the streams from the previous devices and to make the datasets later used in Chapter 5.

Chapter 3 regards synchronization problems between the sensors. Indeed, due to the communication solutions used to transfer the information from the devices to the Pc and the different frame/packet rates, two samples with close timestamps not always refer to the same event. The proposed synchronization algorithm takes into consideration these issues to sort the data in the Pc where will be run the data fusion applications. The second part of the chapter refers to the spatial mismatch between the RGB and depth frames in Kinect. It is caused by the different positions of the cameras in the device and can be solved with a calibration procedure.

Chapter 4 concerns the self-organizing algorithms i.e. unsupervised machine learning solutions that follow the fundamental principles of the self-organization to fit a neural network model to the input data. Three algorithms are introduced: Self-Organizing Map (SOM), Extended SOM (SOM_Ex) and Growing Neural Gas (GNG) networks. The most relevant steps behind each algorithm will be explained and the most important differences between the solutions will be highlighted. The algorithms will be exploited in the last application proposed in Chapter 5, to provide in output an adapted skeleton model useful to monitor the person's movements.

Chapter 5 describes three applications that solve different problems in the AAL framework. The first one is a fall detection algorithm that exploits the depth frame provided by Kinect V1 placed on the ceiling. An alternative solution uses the skeleton in output from Kinect V2 in frontal view and accelerations sampled by two SHIMMER devices set to the wrist and to the waist of a person. The synchronization algorithm allows to use both the data at the same time in a data fusion application able to recognize an eventual risk situation, like a fall.

The second application uses the skeleton from Kinect V2 and a single wearable device to measure kinematic parameters of the body during the TUG test.

The last application monitors the movement performed by a person during intake activities to classify particular actions, such as drinking. In this case, the algorithm uses both RGB and depth streams of Kinect V1 in the same configuration of the first application.

Chapter 6 reports the conclusions concerning the work and the publications list where the

contributions of this thesis are published. Future research directions are discussed and finally possible technological trends of the sensors are outlined.

Chapter 2.

Camera Sensors and Wearable Devices: Principles and Characteristics

This chapter introduces the devices that will be used for the applications proposed in Chapter 5. They can be separated in two principal groups: colour/range cameras and wearable sensors. Initially, the three most important techniques to calculate the distance information (Stereo Vision, Structured Light and Time Of Flight) will be described and then will be introduced Kinect V1 and V2. With regard to the group of wearable devices, the SHIMMER sensor has been selected for its properties of robustness and reliability which will be described to explain how to use the raw data provided by the embedded accelerometer.

Finally, two tools used to handle the raw streams in output from the three different devices will be presented.

2.1. Range Imaging Solutions

This section describes the principles behind the most important optical techniques to calculate the depth data as complement information to the most well-known colour stream. The distance perception is a useful information that can be exploited by a system to improve the knowledge of the 3D environment around it.

Figure 2.1 summarizes the family solutions to estimate the range information using non-contact methods. On the left side, the passive solutions use the radiation already present in the scene to calculate the depth without energy emitted for sensing purpose. The most widely known passive solution is the stereo vision, i.e. the same principle used by the human vision system to "perceive" the depth.

On the other hand, the active solutions are based on a self-generated illumination pattern to find the distance. Light coding and Time Of Flight (TOF) belong to this group and are the technologies respectively implemented in Kinect V1 and Kinect V2. Before the description of these techniques, the Pinhole model is introduced to mathematically represent an ideal camera.

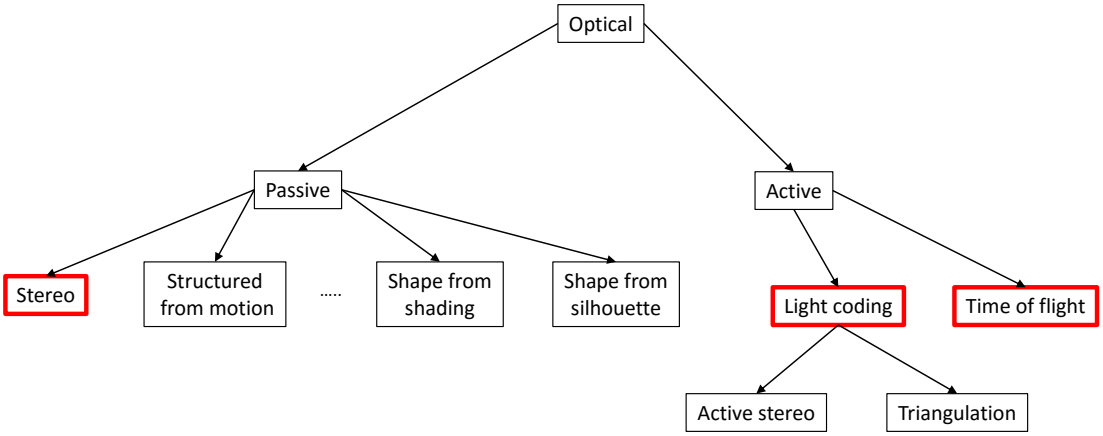


Figure 2.1.: General scheme of the most important range imaging techniques used to calculate the depth data.

2.1.1. Pinhole Camera Model

The Pinhole model is used to represent an ideal camera, as a "dark room", where all the rays of light cross a little hole and "project" an image into the image frame. The model relates a point $P = [x, y, z]$ in the 3D space with the corresponding point $p = [u, v]$ in the frame (Figure 2.2a). The first point is in the Camera Coordinate System (CCS) centred in O (centre of projection), while the second one is in the image plane or Frame Reference System (FRS). This plane is parallel to the x-y plane and placed on the negative z-axis. The distance between O and the image plane is called focal length and the intersection point, called principal point, has coordinates (c_x, c_y) in the image plane. The z-axis is called principal ray or optical axis and connects O with the image plane.

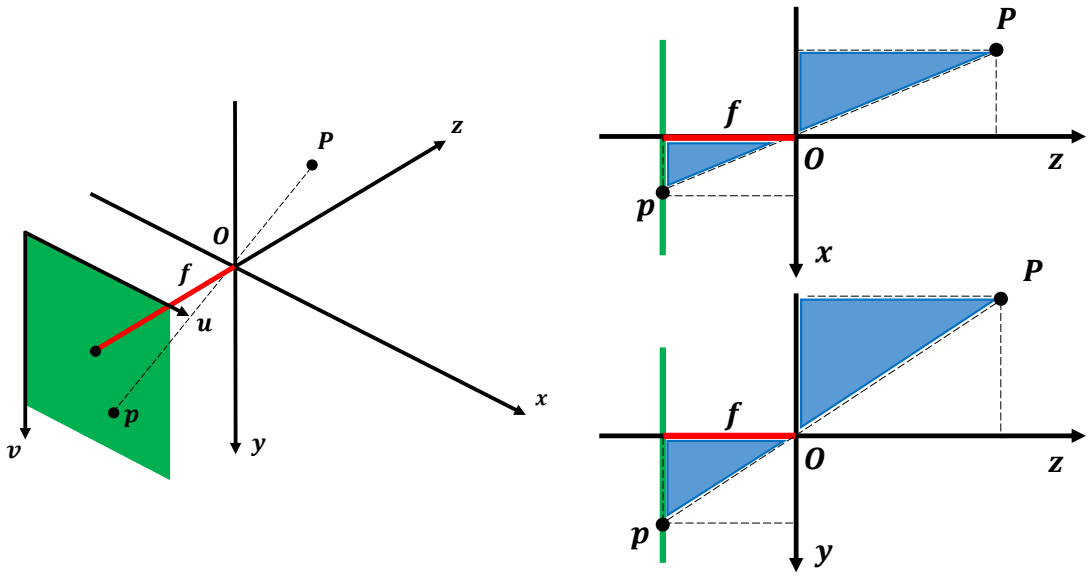
The (u, v) coordinates can be calculated from (x, y), as follows:

$$\begin{aligned} u &= x + c_x \\ v &= y + c_y \end{aligned} \tag{2.1}$$

This is due to the origin of the frame system that starts from the top left corner instead of its centre. In Figure 2.2b are visible the side views of the model parallel to the x-z and y-z planes, with similar triangles depicted in blue. The properties of similar triangles let the following considerations (Figure 2.2b):

$$\frac{u - c_x}{f} = -\frac{x}{z} \qquad \frac{v - c_y}{f} = -\frac{y}{z} \tag{2.2}$$

Placing the image plane in front of the centre of projection and using the homogeneous coordinates,



(a) (b)

Figure 2.2.: Pinhole model. 3D view (a) and lateral views (b).

Eq. (2.2) could be rewritten in the matrix form:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \lambda \tag{2.3}$$

This relation is called image transform and it is worth noting that, inverting the system and setting $\lambda = z$, it is possible to calculate the 3D point in CCS from its pixel coordinates. In practice, the sensor in a digital camera is built in CMOS (complementary metal oxide semiconductor) or CCD (charge-coupled device) technology and the coordinates are in discrete values (pixel). Eq. (2.3) is modified to take into account this information as follows:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

where K is called intrinsic camera parameter matrix or projection matrix and again, λ is a constant. $f_x = \alpha_x f$ and $f_y = \alpha_y f$ are used to consider physical characteristics of the sensor that lead to rectangular pixels instead of square elements, with f in [mm] and α_x/α_y ("aspect ratio") in [pixel]/[mm].

The lens distortion is another physical feature that must be considered in the model represen-

tation of a digital camera. The lens is an object placed in front of the sensor used to increase the amount of light rays captured by the system in order to reduce the time to realise the image. However, it introduces a distortion on the real optics because now light rays do not cross the Pinhole in straight line, but they are curved by the lens. Consequently, the real point in the frame does not satisfy Eq. (2.4) and an anti-distortion model must be used to obtain correct pixel coordinates. The lens distortion can be divided into two components: the radial distortion and the tangential (or decentering) distortion. The first one is due to the shape of the lens and creates a distortion in the border of the frame, which is called "barrel" or "fish-eye" effect. The second arises from the assembly process and leads to a not parallel lens than the image plane. A popular correction model, proposed by Heikkila et al. [92], is:

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} \overline{u}_d \\ \overline{v}_d \end{bmatrix} + \begin{bmatrix} 2p_1 \overline{u}_d \overline{v}_d + p_2(r^2 + 2\overline{u}_d^2) \\ p_1(r^2 + 2\overline{v}_d^2) + 2p_2 \overline{u}_d \overline{v}_d \end{bmatrix} \quad \begin{matrix} r^2 = \overline{u}_d^2 + \overline{v}_d^2 \\ \overline{u}_d = u_d - c_x \\ \overline{v}_d = v_d - c_y \end{matrix} \tag{2.5}$$

Where (u_c, v_c) are the corrected or correct pixel coordinates used in Eq. (2.4), whereas the observed pixel coordinates are indicated with (u_d, v_d). $k = [k_1 \ k_2 \ p_1 \ p_2 \ k_3]^T$ is the vector that stores the distortion coefficients. k_1, k_2 and k_3 are used to compensate the radial distortion and the other two ($p_1 \ p_2$) tackle the tangential distortion. This is not the general distortion model, that uses six parameters for the radial distortion because usually, if the sensor is a standard camera with no wide-angle (for example the "fish-eye" camera), the last three radial distortion coefficients can be ignored. The matrix K and the vector k are the only two elements that must be known to represent a digital sensor without ambiguity. These parameters can be found with a calibration procedure [93, pp. 370–403].

2.1.2. Stereo Vision System

It is the most widely used passive solution that calculates the depth information starting from the frames provided by two cameras. The sensors are placed on a horizontal line and monitor the same area. Specific features are recognized in both the frames and the distance (Z) between the system and the object is calculated thanks to the triangulation principle. The cameras must verify temporal and geometrical constraints. In particular, the two RGB streams must be synchronized.

Before using the triangulation principle, the frame must be rectified and aligned. These operations allow having a specific feature at the same row for both the corrected frames. It drastically reduces the computational time requested to find the feature because now the system has only to scan a single row. In details, as visible in Figure 2.3a an object (chessboard) is captured by both the camera. The distortions are corrected with Eq. (2.5) and then rectified. Therefore, using mathematical transformations, the two frames are placed in the same projection plane and their optical axes are exactly parallel. Finally, the image is cropped to highlight the overlapped regions in both the left and the right images. The theory behind these operations is called Epipolar Geometry [94].

The techniques used to find the same feature in both the left and right corrected images (corre-

spondence problem) are based on similarity measurements and could be divided into three principal groups: local, global and intermediate solutions. In the first group, starting from the coordinates of a point in the left frame, the similarity is calculated in the same row of that point, but in the right image. The second idea uses the opposite approach calculating the corresponding point through an optimization solution that minimises a cost function using a Markov Random Field model. The third category continues using a cost function, but now it is calculated not for the entire frame, but in subset regions [95].

In Figure 2.3b, after the cropping operation, the result is visible and the view is orthogonal to the x-y plane, as well as the frames. The left camera has a CCS centred in O_L and usually it is called reference camera, while the other one is centred in O_R and it is called target camera. The frames have the origin of the horizontal axis centred in u_L^0 for the left frame and u_R^0 for the other one. A difference in the coordinate systems between the stereo solution and Pinhole model (Figure 2.2a) is that now the CCS is behind the reference frame. However, there are no consequences in the mathematical relationship used to find the depth information. The projection of a 3D point (P) can be found in both the processed images at the same row. These points are called conjugate. In the left frame, the point has coordinates $p_L = [u^l, v^l]$ while in the right one it is placed in $p_R = [u^r, v^r]$. With regard to the previous considerations, v^l is equal to v^r. B is called baseline and is the distance between the cameras.

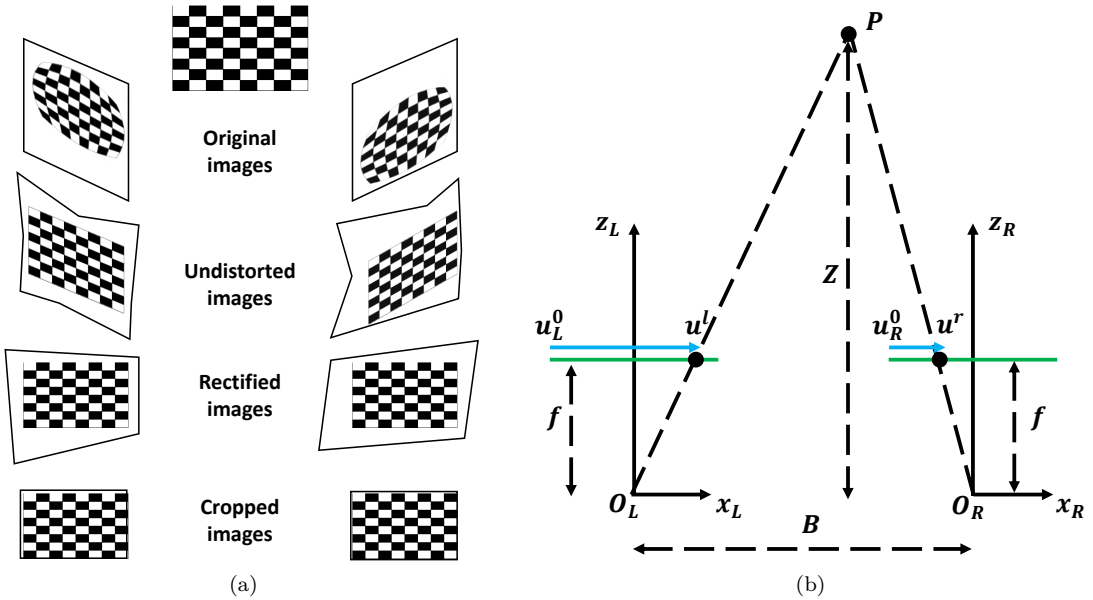


Figure 2.3.: Frame corrections (a). Triangulation with a pair of rectified and aligned cameras (b).

In the ideal case, the two cameras have the same intrinsic parameters and then the same focal length. Z is the distance between the object and the camera system. The base (b_{trl}) of the triangle

with vertexes $u^l P u^r$ can be calculated as follows:

$$b_{trl} = B - (u^l - \frac{WF}{2}) - (\frac{WF}{2} - u^r) = B - (u^l - u^r) \tag{2.6}$$

where WF is the width of the frame.

As visible in Figure 2.3b, the similarity between the triangles $u^l P u^r$ and $O_L P O_R$ is:

$$\frac{B}{Z} = \frac{b_{trl}}{Z - f} \tag{2.7}$$

Substituting Eq. (2.6) in the previous relation, the result is:

$$\frac{B}{Z} = \frac{B - (u^l - u^r)}{Z - f} \rightarrow Z = \frac{fB}{u^l - u^r} \tag{2.8}$$

This operation is called re-projection and the result is the distance Z between the object and the system. Z depends on the focal length of the camera (f) and on the difference between the two horizontal coordinates of the projected points ($u^l - u^r$). This difference is called *disparity* and it is inversely proportional to Z with a non-linear relationship. There is high depth resolution for objects close to the cameras. On the other hand, when two *disparity* values are close to 0, there is a big difference in depth.

If the baseline is large, the accuracy of the depth measurement is improved, especially for objects in the far range. Indeed, for the same images resolution and distance of the observed object from the system, a greater baseline improves the *disparity* resolution. The negative consequence is that the space between the cameras increases. Therefore, the baseline length is a trade-off that should not be ignored when there are some constraints in the system dimensions.

The most important limitation to this solution is that the feature detection issue becomes challenging when the scene is poorly lit or when the monitored area is uniform and it is not possible to find the key-points. For example, if the scene has no variations in colour or geometry like a white wall. For this reason, the stereo vision technique has been formulated as an "ill-posed" problem due to the difficulties to solve the matching issues. Another problem that affects this solution is the presence of occlusions due to the different sensors' point of views. In particular, in Figure 2.3a after the crop operation, the two images frame the same area, but it is possible that some scene portions are not visible at the same time by both the cameras. Finally, the ambiguity is another limit since, in the target frame, there could be more than one feature having same characteristics of a single point in the image framed by the reference camera.

2.1.3. Structured Light

It is an active stereo vision technique that overcomes the limits of the previous approach. In particular, the ambiguity problem can be reduced if a specific pattern is used to find the conjugate points inside the images. To implement this solution, one of the sensors is substituted by a projector that illuminates the scene with a pre-defined pattern. The other camera catches the reflected pattern and, through the correspondence estimation process, interprets the captured frame to find

the disparity. Therefore, Eq. (2.8) is still valid to calculate the depth information and now the baseline is the distance between the projector and the observed camera. The Structured Light solution can be also implemented with more than one camera and in this way the ambiguity could be even further limited, but the system costs increase.

In case of a white wall scene discussed in the previous section, the projector illuminates the area with a specific pattern. Differently from the stereo system, now the receiver is able to find the corresponding point associated to the emitted ray. For each pixel of the projected pattern there is a corresponding *code-word*. This means that a local pattern distribution is assigned to a projected pixel. The characteristics of the chosen pattern distribution must be carefully selected. In particular, it can be implemented with different illumination values assigned to each pixel. Starting from binary scheme or even more complicated ones that use grayscale or RGB scale. The pattern distribution can also exploit more than one pixels per code-word.

The projector illuminates the scene with a pattern that is the sum of all code-words assigned to all pixels in the emitted frame.

Among all the possible combinations of these two implementation schemes (illumination, area) a subgroup is selected and it represents the code-word alphabet. The most widely used schemes for the projection pattern [96] are the following:

- *direct coding*: every single pixel of the projected pattern is assigned to a single value such as grayscale level or RGB colour [97]. The cardinality of the code-words has the same dimension of the projected pattern;

- *time-multiplexing coding*: the code-word is assigned to each pixel as a sequence of intensity values that change during the time. Therefore, the emitted information is not projected at single time, but it is a combination of different patterns;

- *spatial-multiplexing coding*: a single code-word is now associated to a window of pixel centred in a specific point. Consequently, the code-words are not independent because they have some pixels in common [98].

Unfortunately, both the transmission and acquisition processes can alter the projection pattern and, when this happens, the consequence is a wrong estimation of depth information. The causes are:

- *perspective distortion*: neighbour pixels in the projected pattern are mapped in far pixels in the captured frame due to different distances among the objects in the scene;

- *colour distortion*: the intensity level of the captured frame can be locally altered by different reflection properties of the objects in the scene;

- *external illumination*: the sunlight or other light sources can change or even saturate the intensity level of the captured frame;

- *occlusions*: the emitter and the camera are physically placed in two different points of the system; it could happen that some pixels in the projected pattern are not visible to the camera. The shadow area is an example of this problem, i.e. a portion of the projected pattern cannot be captured by the camera as it is hidden by an object's shadow;
- *real camera and projector*: the emitter and the receiver are real objects which introduce non-linear behaviours.

These distortion sources could affect, with more or less strength, each one of the three coding schemes previously introduced. In particular, the direct coding solution is influenced by changes in colour intensity due to external illuminations, but it does not suffer the dynamic changes in the scene (such as when an object is moved in the area). On the other hand, the time-multiplexing coding is able to take into account the grayscale changes, but the sequence of consecutive patterns belonging to the same code-word must be faster projected than the object displacement. The third scheme is the most complete because it needs only a single projected pattern and it is robust to colour distortions. However, the size of the window must be accurately selected since it is a trade-off between robustness to occlusion (small window) and cameras noise/colour distortion (big window).

When the *correspondence problem* cannot be solved and for a pixel of the captured image there is not a conjugate element in the projected pattern, the depth information for that coordinate is set to a specific value such as zero.

2.1.4. Time Of Flight

The TOF principle is not new, it has been used in the last years for a plenty of different applications such as spectrometry, SONAR and spectroscopy. Recently, thanks to the hardware's improvements and to the reduction of the costs, it started to be exploited in the consumer area too.

The idea behind the TOF camera is quite simple: an emitter lights an area and the transmitted photons are reflected by the objects in the scene. The receiver catches the reflected photons and, analysing the photons' speed as well as the transmitted/received time interval, calculates the distance between the sensor and the objects. The cameras that exploit this solution are called direct TOF.

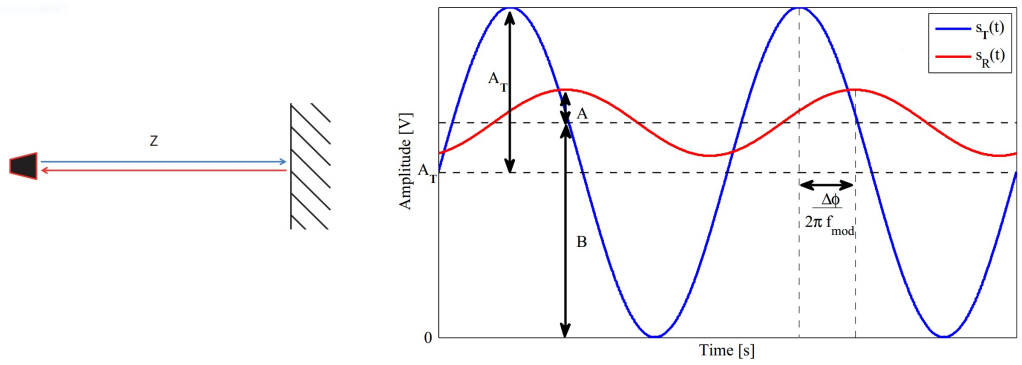
In particular, a direct TOF implements the following elementary equation:

$$Z = \frac{ct}{2} \tag{2.9}$$

where $c = 3 \times 10^8\, m/s$ is the photons speed, Z is the distance between the sensor and the object and t is the required time by the photon to complete the path (Figure 2.4a). Finally, the factor 2 is used to consider half of the total path from/to the emitter. This solution is exploited for spot measurements and, to let the camera to calculate the distance of a wide surface, the emitter must move in vertical and horizontal directions. The LIDAR system (Light Detection and Ranging) is a possible application of this technology. Considering Eq. (2.9), the photon needs 6.6 ps to travel 1 mm. Consequently, to have a depth accuracy of 1 mm it is necessary a clock system with resolution

close to 5 ps. The result is higher costs for the clock component of the sensors that use the direct TOF principle [99].

The indirect TOF technology is an alternative solution that uses the light modulation to replace the direct time measure in Eq. (2.9). In particular, in the *Continuous Wave Intensity Modulation* (CWIM) [100] technology, the range information is coded is the phase difference between the transmitted sinusoidal signal and the corresponding reflected one (Figure 2.4b).



(a) (b)

Figure 2.4.: Scheme of the direct TOF (a). Transmitted and received signals in a CWIM solution [95].

Let $s_T(t)$ be the transmitted signal by a near-infrared (NIR) illuminator:

$$s_T(t) = A_T[1 + \sin(2\pi f_{mod} t)] \tag{2.10}$$

where A_T is the amplitude of the transmitted signal and f_{mod} is its frequency. After the reflection, the received signal is:

$$s_R(t) = A_R[1 + \sin(2\pi f_{mod} t + \Delta\phi)] + B_R \tag{2.11}$$

where A_R is the amplitude of the received signal and B_R is the background illumination captured by the receiver. Eq. (2.11) can be rewritten as follows:

$$s_R(t) = A\sin(2\pi f_{mod} t + \Delta\phi) + B \qquad A = A_R \quad B = A_R + B_R \tag{2.12}$$

The receiver calculates the cross-correlation function between $s_T(t)$ and $s_R(t)$ and samples it in four positions. Unknowns A, B and $\Delta\phi$ will be found combining these multiple samples (C_0, C_1, C_2, C_3) [101]:

$$A = \frac{\sqrt{(C_0 - C_2)^2 + (C_1 - C_3)^2}}{2} \quad B = \frac{C_0 + C_1 + C_2 + C_3}{4} \quad \Delta\phi = \arctan\left(\frac{C_0 - C_2}{C_1 - C_3}\right) \tag{2.13}$$

Using Eq. (2.9), the phase term in the previous equation could be exploited to find the distance:

$$\Delta\phi = 2\pi f_{mod}\tau = 2\pi f_{mod}\frac{2Z}{c} \rightarrow Z = \frac{c}{2 \cdot 2\pi f_{mod}}\Delta\phi \qquad Z_{max} = \frac{c}{2f_{mod}} \tag{2.14}$$

Taking into consideration that the $\Delta\phi$ is periodic in $[0, 2\pi]$, there is an interval that, if exceeded, leads to an ambiguous measurement. The maximum distance (Z_{max}) is found substituting 2π to $\Delta\phi$ in the previous equation, and it is related to f_{mod}. For example, for f_{mod} of 20 MHz, the maximum range is 7.5 m. With a low value of f_{mod}, the range of the TOF camera increases, but there is a negative consequence for the depth accuracy [102]:

$$\sigma_d = \frac{c}{4\pi f_{mod}\sqrt{2}}\frac{\sqrt{B}}{A} \tag{2.15}$$

The standard deviation of the depth (σ_d) is also related to A and B. It decreases when the external IR illumination is low or the intensity of the received signal is high. In light of these considerations, the frequency of the modulated signal is a trade-off between the covered area and the accuracy measurement.

Usually for each pixel, a real TOF does not have a couple emitter and receiver, but the system is formed by a single IR illuminator and a CCD/CMOS sensor. For each point of the frame, the previous equations continue to be valid and, for each useful time, the TOF camera provides in output: the depth frame and the amplitude image. The second element is a matrix of A values related to each pixel.

Most of the TOF camera implements also the *Suppression of Background Intensity* (SBI) algorithm. It means that the emitter is on or off according to a specific sequence, whereas the receiver, using a synchronization signal, is able to distinguish if the captured frame has been recorded when the emitter was on or off. Let f_{on} be the frame when the transmitter is on and f_{off} the corresponding one when it is off. The system is able to find the information of the only reflected objects lighted by the sensor ($f_{on} - f_{off}$) [103]. In this way, the amplitude frame is less susceptible than the external light and it represents an improvement with respect to the previous two technologies. Finally, a band pass filter centred in the operative frequency of the emitter is used by the receiver as another solution to reduce the external noise.

Compared to the Structured Light principle, the advantages are:

- a single viewpoint is used, so that this technology is more robust to occlusion (no shadows) and the depth information continues to be available for sharp surfaces. As a consequence of this, there are less zero pixels;

- the external light is discarded from the receiver and this mitigate the interference problem. However, the sunlight can still saturate the received signal and it could compromise the depth information.

2.2. Kinect V1

The first version (Kinect 360), based on the technology developed by PrimeSense [104], was released by Microsoft in November 2010 as an alternative controller for the gaming console Xbox 360. The version for computer applications (Kinect for Windows) was released in February 2012, but there were no hardware differences between the two versions. The sensor uses the Structured Light principle with a spatial-multiplexing scheme as projected pattern to calculate the depth information. The pattern (speckles), generated by a diffraction grating placed in front of the IR emitter, follows a pseudo-random distribution for each row. It leads to a covariance close to zero when calculated between two spatial-multiplexing windows centred in two different pixels of the projected and received pattern. Only if the two pixels effectively correspond to the conjugate couple, the covariance reaches the maximum value. The algorithm behind the spatial-multiplexing scheme, implemented by Kinect V1, is still undisclosed and some reverse engineering analyses found that the correlation window size is 9×9 pixel [100].

To overcome the non-ideal behaviours of emitter and receiver, a reference image [105] is used to calculate the depth. It is saved in the device and refers to an acquisition of a horizontal surface parallel to the camera at a specific distance from the system. Due to the configuration of the reference image (flat surface), the calculated disparity matrix has the same value for all the coordinates of the received image. When a generic scene is captured by the system, it is possible to express the position of each pixel with respect to the corresponding element in the reference image. Indeed, if the generic scene is closer or farther than the reference image, the projected point will move along the same row of the captured image [106].

The most important elements of Kinect V1 sensor are:

- microphone array;
- tilt motor;
- RGB camera;
- IR camera;
- IR projector;
- 3-axes accelerometer.

Figure 2.5 shows the external view of Kinect V1 and the corresponding cameras. The green rectangle encompasses the IR projector, while the red and yellow colours highlight respectively the RGB and IR cameras. The baseline, i.e. the distance between the projector and the IR camera, is approximately 7.5 cm. The microphone array is made up of four sensors that sample the audio with a frequency of 16 KHz and provides in output a digital stream of 24 bit resolution per sample. It is possible to recognize the position of the signal using a beamforming solution. The FOV is 57° and 43° for the horizontal and vertical planes, respectively. A tilt motor can be used to increase the vertical range thanks to a tilt span of $[\pm 27]$ degrees.

The PrimeSense PS1080-A2 chip, using the IR stream and the Structured Light principle, calculates locally the depth before sending the data to the Pc. The component also handles the other streams (IR emitter, RGB, audio) and sends them to a host system by a USB 2.0 connection, after the installation of dedicated drivers [107]. Other third-party software toolkits, such as OpenNI

(a) (b)

Figure 2.5.: External view of Kinect V1 (a) and internal components: IR projector (green), RGB camera (red) and IR camera (yellow) (b).

[46] and Libfreenect [108], have been published to handle Kinect streams not only for Windows platform, but also for both Linux and Mac OS operating systems.

The colour stream is encoded with different resolutions and schemes, such as RGB, YUV and Bayer. The most important one is RGB that stores the pixel information in 32 bit through the scheme X8R8G8B8, where X is an empty channel. The resolutions for the RGB stream are:

- 640×480 @ 15 fps;
- 640×480 @ 30 fps;
- 1280×960 @ 12 fps.

The YUV colour space uses 16 bit per pixel and a single resolution configuration of 640x480 at 15 fps.

Finally, like the RGB, the Bayer scheme stores the pixel information in 32 bit (X8R8G8B8) with the following resolutions:

- 640×480 @ 30 fps;
- 1280×960 @ 12 fps.

In addition to the RGB camera, also the IR sensor provides a colour format for the IR stream. It is a grayscale format in 16 bit where the last 6 ones are set to 0. There is only a resolution setting for the IR stream and it is 640×320 @ 30 fps.

The depth frame stores the distance information from the monitored object to the camera plane of the IR sensor with a resolution of 16 bit. In details, the first 13 bit store the depth information in mm and the other 3 bit identify the player index. Kinect for Windows version could be set in near or default mode. In the first case, the working range is 0.4 – 3 m, whereas in the default mode the range is 0.8 – 4 m. Starting from the Software Development Kit (SDK) 1.6 and later, in the default mode, distance measurement has been extended up to a maximum value of 8 m, even if in this situation the distance error is not negligible [106].

The formats for the depth stream are:

- 640×480 @ 30 fps;
- 320×240 @ 30 fps;
- 80×60 @ 30 fps.

A skeleton engine [45] uses the depth matrix in input to calculate the skeleton model of the person who is framed by the sensor. It is characterized by 20 nodes placed in the most important joints of the human skeleton [109]. The maximum number of skeletons provided by Kinect V1 is

two with a specific player index field in the person's pixels of the depth frame. This information can be exploited to get directly the person's blob in foreground.

When nobody is in the scene, the player index field is set to zero. Whereas, when more than two players are visible in the monitored area in addition to the two skeletons, the engine provides a central point for each other person recognized, for a maximum of 6 subjects tracked at the same time.

Through the Pinhole representation for the IR camera, the joint coordinates can be referred to the CCS (x, y, z) or to the depth frame (u, v, z). In the first case, every joint is a point in the 3D space, whereas in the second one the skeleton is superimposed to the depth frame.

An alternative skeleton engine, called NiTE, was released by OpenNI [46]. Similarly to the algorithm designed by Microsoft, the NiTE provides 15 joints per skeleton with no limits on the maximum number of people tracked at the same time by the system.

Finally, starting from the depth frame and using the intrinsic parameters of the depth camera, the 3D representation of the distance map is found. It is called Point Cloud (PC) and it is calculated inverting Eq. (2.4) and setting $\lambda = z$. Therefore, for every single point (u, v) of the depth frame, and in case its distance is available, exist a corresponding point (x, y, z) in the depth CCS. The PC will be another useful information to be exploited, together with the other streams previously introduced, in the applications described in Chapter 5.

2.3. Kinect V2

Microsoft in Q3 of 2014, during the same presentation event of the gaming console Xbox One, has released the second generation of the Kinect sensor.

Instead of the classic passive approach based on the stereoscopic principle, also for Kinect V2 has been chosen an active solution. However, the similarity with Kinect V1 ends here because, instead of the Structured Light principle, Kinect V2 exploits the TOF technology. In particular, thanks to the partnership with the American TOF maker Canesta, Microsoft chose the CWIM technology and designed itself the range sensor [110].

As remarked in Eq. (2.14-2.15), there is a trade-off between depth accuracy and maximum range. To overcome this limit, Kinect V2 implements a modified version of the CWIM technology. The receiver calculates three cross-correlation functions between the received signal and three different versions of the transmitted signal, modulated at 16 MHz, 80 MHz and 120 MHz, respectively [111]. Then, the cross-correlation functions are sampled three times (instead of four) [112]. In this way, the maximum range of the camera can be extended without loss of accuracy [113].

Figure 2.6 shows the external and the internal views of Kinect V2. The three coloured rectangles highlight the RGB camera, the IR receiver and three infrared beacons. The microphones array is still present and now the sampling frequency is set to 48 kHz, while the tilt motor has been removed. The official/unofficial drivers are available in [114] and [115].

The different streams provided by the sensors are:

– RGB frame with resolution of 1920×1080 (full HD) @ 30 fps;

(a) (b)

Figure 2.6.: External view of Kinect V2 (a) and internal components: RGB camera (red), IR camera (green) IR projectors (yellow) (b).

- IR frame with resolution of 512×424 @ 30 fps;
- depth frame with resolution of 512×424 @ 30 fps and the recommended working range is between 0.5 m ÷ 4.5 m;
- audio (f_c = 48 kHz).

Compared to the other TOF cameras on the market, the resolution of Kinect V2 is higher and available at a price lower of one or two orders of magnitude. However, due to the improvements in streams resolution, the minimum Pc specifications to handle the sensor are more stringent and the communication need a USB 3.0 connection.

In Table 2.1 are summarized the most important characteristics of the two sensors. In Kinect V2 the resolution of the three streams has been modified. Especially, the frame rate of the RGB automatically switches from 30 fps to 15 fps if there are low ambient light conditions. The FOV is changed since in the second generation it is of 70° in the horizontal plane and 60° in the vertical one.

It is worth noting that the maximum resolution of the depth frame of Kinect V1 (640×480) is higher than the corresponding one in the second generation of the sensor (512×424). Anyway, in Kinect V1 the depth information is obtained from the interpolation of a depth matrix. Indeed, the receiver, to correctly find the distance information, needs the points in the emitted pseudo-pattern spaced of a minimum value. Therefore, only a pixel every 20 is effectively calculated by the Structured Light principle [103].

Different studies have compared the performances of the two sensors and have shown an improvement in the depth accuracy for Kinect V2 [111, 116]. In addition, Kinect V2 is able to calculate the depth information, inside a range of 1.5 m, also when it is used outside with direct sunlight [117].

Another important difference from Kinect V1 is the number of sensors that can be connected to the Pc. For Kinect V2, only one device can be connected to the Pc, but multiple applications can exploit the sensor at the same time. On the other hand, when an application starts the communication with Kinect V1, it holds the device until the program is closed.

Finally, the software engine that provides the skeleton has been improved too [118]. The number of joints has been incremented to 25, adding the points that model the neck, both the left and the right thumbs as well as the final part of the palm. The total number of tracked people move from 2 to 6.

Features	**Kinect V1**	**Kinect V2**
Depth Sensing Technology	Triangulation with Structured Light	Time of flight
Colour Image Resolution	640x480 @ 15/30 fps 1280x960 @ 12 fps	1920x1080 @ 30 fps (15 fps low light)
IR Image Resolution	640x480 @ 30 fps	512x424 @ 30 fps
Depth Sensing Resolution	640x480 @ 30 fps 320x240 @ 30 fps 80x60 @ 30fps	512x424 @ 30 fps
Field of View	43° vertical 57° horizontal	60° vertical 70° horizontal
Depth Sensing Range	0.4 m – 3 m (near mode) 0.8 m – 4 m (normal mode) Max: Up to 8 m (tested)	0.5 m – 4.5 m Max: Up to 8 m (tested)
Skeleton Tracking	Up to 2 subjects 20 joints per skeleton	Up to 6 subjects 25 joints per skeleton
Built-in Gestures	None	Hand state Hand pointer controls
Microphones	yes	yes
Unity Support	Third party	yes
Face APIs	Basic	Extended
Runtime Design	Can run multiple Kinect sensors per computer one app per device	One Kinect per Pc Multiple apps share same device
Windows Store	No	Yes
Minimum Hardware Requirements	Dual-core 2.66 GHz 2 GB RAM Direct X9 sup. USB 2.0 host controller	64 bit (x64) i7 2.5 Ghz 4 GB RAM Direct X11 sup. USB 3.0 host controller
Software Requirements	Visual Studio 2010	Visual Studio 2012/13

Table 2.1.: Most important differences between Kinect V1 and V2.

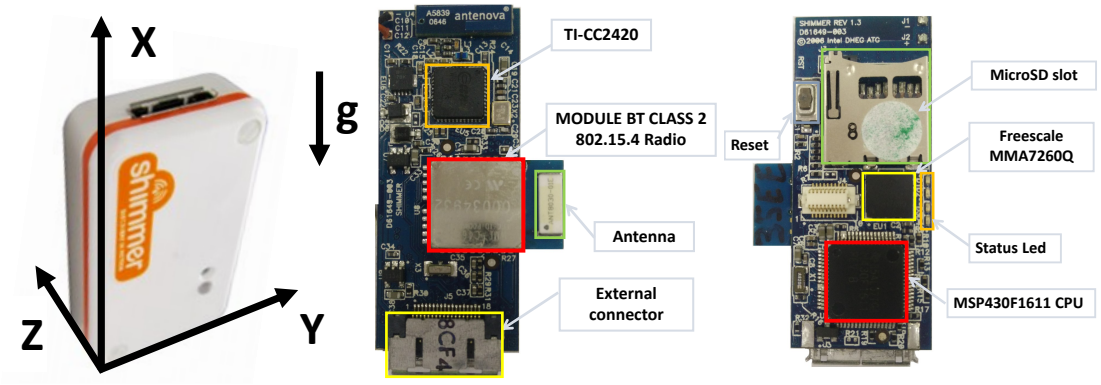
2.4. Wearable Device

The elderly activities can be monitored not only with ambient sensors, but also with wearable devices. For example, to mention a few: MICAZ, TelosB, Imote2. When the aim is to monitor the healthy condition of a person, the SHIMMER [119] device is the most appropriate solution. The mean of the acronym is Sensing Health with Intelligence, Modularity, Mobility and Experimental Reusability.

The most important characteristics that have influenced the choice of this platform are the following:

- embedded peripherals;
- expansion modules (ECG, gyroscope, daughterboard);
- low energy system;
- wireless communication using Bluetooth (BT);
- internal storage;
- free licence programming language.

The external view of SHIMMER Rev. 1.3 is visible in Figure 2.7a, while in the other images are highlighted the most important components of the device.



(a) (b) (c)

Figure 2.7.: External view of the SHIMMER Rev. 1.3 and orientations of acceleration axes (a). Frontal (b) and back (c) views.

Table 2.2 summarizes the most important features of SHIMMER Rev. 1.3.

Thanks to the TinyOS operating system, it is possible to change the functionality of the device. For example set the operative range of the sensor MMA7260Q, control the transmission rate of the BT packets, switch on/off the LEDs, etc. The low energy characteristic, considering a battery capacity of 250 mAh, allows reaching an autonomy of 10 days with a sampling frequency of 100 Hz of the accelerometer and a regular radio communication [120].

In the applications presented in Chapter 5, the information provided by the embedded ac-

Features	Description
I/O	– Three axis accelerometer Freescale MMA7260QT 1.5/2/4/6 g – 4 coloured LEDs – Reset button – 12 general purpose connectors
Microcontroller	– MSP430F1611 CPU 10 Kbyte RAM, 48 Kbyte Flash Clock 8 MHz 8 channels A/D (12 bit) Low energy chip
Memory	– MicroSD slot 2 GB
Communication	– Chipcon CC2420 BT class 2
Form factor	– 1.75” × .8” × .5” and 10 grams without BT
Operating system	– TinyOS

Table 2.2.: General properties of SHIMMER Rev. 1.3.

celerometer with a resolution set to ± 4 g and sampled at 100 Hz has been exploited. The axis orientations are visible in Figure 2.7a. The raw data $(x_{raw}, y_{raw}, z_{raw})$ are converted to g acceleration (X, Y, Z) using Eq. (2.16). In this way, when an axis of the accelerometer is parallel to the gravity vector and there are not external factors affecting the measurement, the acceleration will be 1 g. Whereas, if the same axis is in the orthogonal configuration, the value will be 0 g.

$$X = \frac{x_{raw} - x_{0g} + x_{bias}}{V_{sens}} \qquad Y = \frac{y_{raw} - y_{0g} + y_{bias}}{V_{sens}} \qquad Z = \frac{z_{raw} - z_{0g} + z_{bias}}{V_{sens}} \tag{2.16}$$

The accelerometer provides in output a voltage value in the interval 0-3.3 V, proportional to the acceleration. Then, the value is associated to a discrete number in the interval [0 4096] by the 12 bit A/D converted, as indicated in Table 2.2. The 0 g quantity is measured by the device when the specific acceleration axis is perpendicular to the gravity force. For example, in the situation visible in Figure 2.7a, the Z and Y axes are in the 0 g configuration and their values in output from the A/D converter can be substituted in Eq. (2.16), respectively to y_{0g} and z_{0g}. Thus, the scale of the accelerometer is now centred in 0 g.

As indicated in the datasheet of Freescale MMA7260QT [121], the accelerometer, when set in the range ± 4 g, has a sensitivity (V_{sens}) of 300 mV/g. This quantity, converted to the corresponding digital value (372), is used as denominator of Eq. (2.16) to find the acceleration in the g scale. Finally, x_{bias}, y_{bias} and z_{bias} are weight coefficients calculated through a calibration procedure which takes into consideration the nonlinearities of the sensor. Indeed, there is a mismatching between the ideal value and the information provided by the sensor. In particular, in the 0 g configuration the sensor should measure information close to 2047, but in practice there is a bias, the weight coefficients are used to compensate this difference.

The acceleration magnitude is calculated combining all the components in the three axes:

$$M_{acc} = \sqrt{X^2 + Y^2 + Z^2} \tag{2.17}$$

Finally, the acceleration information can be used to get the orientation of the sensor (in radiant) than the gravity direction [28]. In particular, the equations are:

$$\theta_x = atan2\left(\sqrt{Z^2 + Y^2}, X\right) \quad \theta_y = atan2\left(Y, \sqrt{X^2 + Z^2}\right) \quad \theta_z = atan2\left(Z, \sqrt{X^2 + Y^2}\right) \tag{2.18}$$

For example, for the configuration in Figure 2.7a, the orientation angles after the conversion from radiant to degree, are $\theta_x = 180°$, $\theta_y = 90°$ and $\theta_z = 90°$.

2.5. Recording Tools

This paragraph introduces the recording tools used to store the streams provided by Kinect V1 and V2 as well as the data from the wearable devices. They are implemented in C++ language using the Integrated Development Environment (IDE) Visual Studio 2010 for the first tool and the 2012 version of the same IDE for the second program. The SDKs, used to handle the streams

provided by the devices, are the official ones realised by Microsoft, SDK 1.8 for Kinect V1 and SDK 2.0 for Kinect V2.

The second part of the paragraph describes the datasets built with the two programs, published in [122] and used as input data to the algorithms described in Chapter 5.

The *Kinect Studio* is the official program realised by Microsoft to store the information provided by the cameras. The data are saved in .xed file extension, a proprietary format that can be opened by the same application only. In addition, only the depth, RGB and IR streams can be recorded while there is no possibility for the skeleton joints. An alternative approach is the ONI format proposed by OpenNI, as it saves the video streams as a single file. Differently from *Kinect Studio*, now the file can be loaded by any application that uses the OpenNI SDK, because the program handles the file as a virtual device connected to the Pc. However, both the .xed and .ONI files cannot be opened directly by third party applications, which are written in program languages not supported by the two SDKs.

A possible solution is to use directly the APIs of the SDKs and design a program that can handle and save the frames. The official SDK allows two types of programming languages: managed (C# or Visual Basic) or unmanaged (C++) code. In the first case, the framework used is ".NET", whereas in the second group the SDK allows a set of native functions. For the designed applications, the second type program language has been used considering the speed requirements.

Different tools directly using the official SDK, are on-line published [123]. The most complete one is the *Kinect Stream Saver Application* [124]. It stores both the streams provided by Kinect V1 and the timestamp assigned by the SDK to the samples, but the application has some drawbacks. For instance, some frames could be lost during the recording process. This is due to the solution implemented to save the streams. It is based on a multi-thread approach where a dynamic FIFO buffer is used to handle and save the frame at the same time. Therefore, its performances are related to the speed capabilities of the storage device. In addition, the structure used to store the recorded frame does not separate the image each other, but combine all the information in a single file.

Other tools are available [125, 126, 127, 128] for Kinect V2, but not all the streams can be saved or some limits are present. For example, the program in [128] is able to record all the streams, but also in this case, some frames could get lost during the recording process. In particular, the probability that an RGB image is not saved increases dramatically due to the considerable size of the frame (7.91 MB).

The most important steps to handle a sensor stream are:

- initialize the device (get the sensor handler) and open the communication;
- initialize the stream to read;
- catch the event of new frame received using the dedicated routine;
- read the data and release the stream;
- close the communication with the sensor when the program is closed.

When a stream is open, its data is stored in the RAM as a single array, starting from a specific

memory address to the next bytes for the entire dimension of the image. The program reads the region and saves the frame information in a local buffer.

As introduced in Section 2.2, the possible streams provided by Kinect are colour, depth, IR, skeleton and audio. Each one is controlled by a specific function. The audio source is the only stream that is not handled by these tools because this information is not used in the designed applications.

Figure 2.8 shows the first tool (*Complete Viewer V1*), three boxes are used to display the different streams. It means that three streams are enabled after the device initialization. The first one regards the depth and in this specific situation, the person's pixels are in blue colour because the skeleton engine has recognized a person inside the monitored area. The other useful pixels are in grayscale colour, while the black colour is used to represent the areas where the distance information is not available.

In the second window, the tracked skeleton is visible and each node is a joint in the depth FRS. The green segments are only used to have a better representation of the human model. Finally, the last box refers to the colour frame at a resolution of 320×240 pixel. One important difference between the two official SDKs is that the IR stream is seen by Kinect V1's library as a particular type of colour stream and subsequently it is not possible handling IR and colour streams simultaneously. Therefore, in Figure 2.8 it is not visible the IR stream.

The combo box called *Kinect* (violet rectangle) is used to select the device, indeed more than one Kinect sensor can be connected to the Pc at the same time.

The possible streams to be saved are:

- depth frame;
- colour frame or IR frame;
- skeleton;
- mapping matrix (available if depth and colour streams are open).

The RGB and depth cameras are placed in different positions inside the Kinect device and, consequently, the two raw frames cannot be directly superimposed. The mapping matrix is a lookup table used to map RGB frame in the depth image. In particular, when the resolution of the depth and RGB frames is 320×240, the matrix has 640×240 elements. Each row has 640 entries that represent 320 couples $(row, column)$. Each depth pixel (u_D, v_D) has the associated RGB pixel coordinates stored in the mapping matrix at the position $(u_D, 2v_D)$. There is a mapping matrix for each couple of depth-RGB frames. More information regarding the mapping theory will be introduced in Section 3.2.

The combo box called *Color view* (red rectangle) is used to switch between the RGB or IR streams and *depth resolution* sets the frame size to one of the possible resolutions described in Section 2.2. The combo boxes called *Max depth/skeleton/RGB Frames* (green rectangle) are used to select the maximum number of frames to be saved between 100 or 1000. For each stream, the user has two possible alternatives. For example for the depth stream, he can record a single image every time the button *Capture depth view* is pressed or he can store a sequence of frames clicking the button *Start Capture Depth View* (yellow rectangle). The result is the same for the corresponding *Skeleton* and *RGB* sections.

Figure 2.8.: Recording tool for Kinect V1.

Finally, the button *Start Capture Skeleton + Depth + Video Data* (blue rectangle) is used to record three streams at the same time. Each depth frame is stored in a separate binary file, whereas the RGB or IR images use the bitmap format. They are saved in the same folder where there is the executable file of the recording program. The combination one-file one-frame is useful when a single element must be analysed without the need to read all the stored files as occurred in [124]. Whereas, due to the negligible size of the joints structure, all the skeleton coordinates are saved in two files; one in frame coordinates and the second in depth CCS, both of them in txt or csv formats.

The program starts saving the skeleton coordinates with the depth frames and two people can be tracked with 20 joints coordinates per skeleton. The joints are saved also with the player index and the properties of the point (*tracked* or *inferred*). If no people are inside the monitored area, the coordinates are set to 0 and the other two fields are filled with special values.

Two timestamps are calculated for each stream and correspond to two different times:

- SDK local time;
- Query performance counter (QPC)[129], local time from the operating system.

The first one is the time reference (in ms) provided by Microsoft SDK and it starts when the sensor is initialized [130]. The QPC is calculated from the total number of ticks that have occurred since the operating system was started, with a resolution of 1 μs.

Complete Viewer V2 is the name of the recording tool for Kinect V2, published in [122]. Figure 2.9 shows the GUI of the program. The appearance is similar to the first tool but there are some differences. The different FOVs of the two cameras are noticeable, indeed even if the two sensors

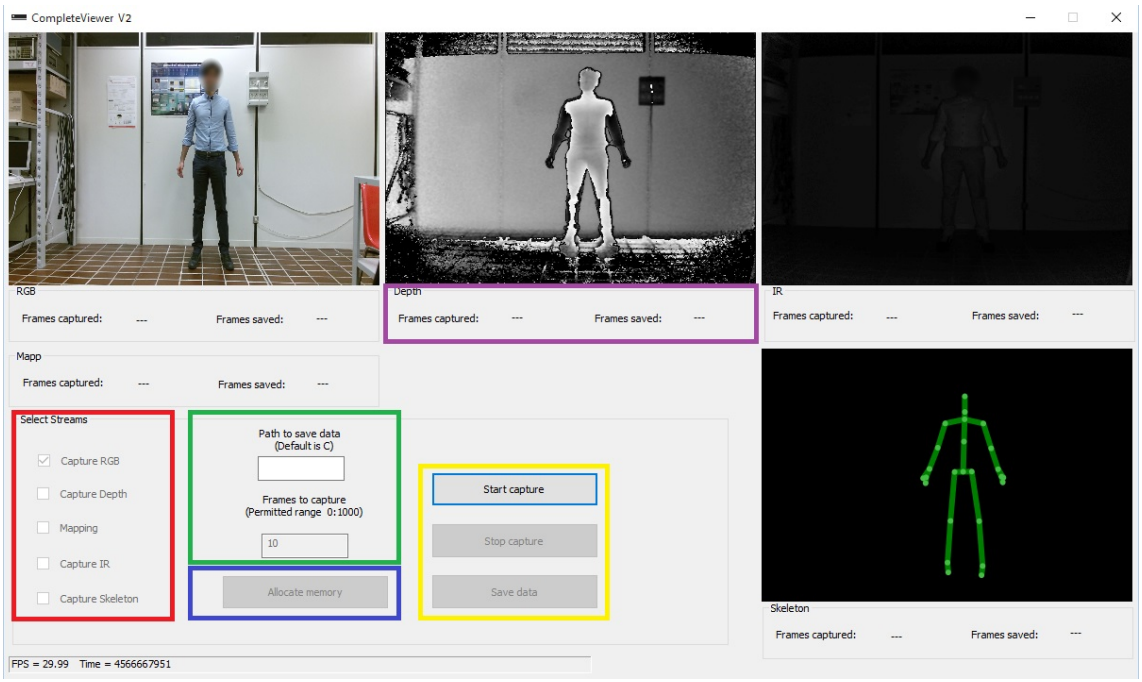


Figure 2.9.: Recording tool for Kinect V2.

are very close, the area monitored by Kinect V2 is bigger. Both the IR (third box) and colour (first box) streams are now opened and used at the same time. The official SDK 2.0 can handle only a single Kinect V2 device connected to the Pc, therefore the corresponding combo box to select the device has been deleted.

The possible streams to be saved are the same described for the first device, but now different combinations are possible thanks to the check boxes called *Select streams* (red rectangle).

The mapping matrix check box becomes available when the depth and colour frames are selected. It has the same meaning described for Kinect V1. The difference is in the number of elements that now are 1024×424.

In addition, using the dedicated text box, now the number of frames that can be saved is not limited to two values, but is configurable between a range going from 1 to 1000. It is also possible to select the path in the Pc where saving the captured streams. To this end, the user needs to specify the path in the text box called *Path to save data* (green rectangle), otherwise the application will use the standard one. For instance, in Figure 2.9 the user has selected 10 frames of the RGB to save in C:\\. After the choice of streams and path, the button *allocate memory* (blue rectangle) reserves a portion of the RAM where the selected streams will be temporary allocated. The required space is dynamically calculated at run-time, considering the type of streams selected and the number of frames to be saved. This step can take some time and for this reason, it has been separated from the recording phase. Otherwise, when the user presses the button to record (*start capture*), the program spends some time to allocate memory and initial frames could not be saved.

After that, pressing the button *start capture*, the program begins allocating the information. The button *stop capture* stops the recording before all the allocated memory is filled. Finally, with the button *save data* all the information can be stored in the non-volatile memory, ready to be used as input by other applications. These three buttons are highlighted with the yellow rectangle.

This final step will be executed when all the selected streams are stored in the RAM because it could reduce the system performance. Indeed, if the data is saved in the hard disk while the application is recording, the probability that some frames are not handled by the program increases, generating a direct reduction of the frame rate, i.e. the same issues that affect the tools in [124, 128]. To increase the speed whereby the streams are saved, a dedicated thread is thrown.

In Figure 2.9, below each stream, there are two labels called: *frame captured* and *frame saved* (violet rectangle). When the button *start capture* is pressed, the first label displays the number of frames stored in the RAM, whereas the other label shows the data stored in the hard disk.

Inside the chosen path, a specific folder is created for each stream. The file formats are the same described for Kinect V1 and the only difference is in the maximum number of the tracked people (6) and the total joints per skeleton (25).

When Kinect V2 and SHIMMER are used together for data-fusion applications, both the data must be recorded at the same time. In particular, a wireless communication between the wearable device and the Pc is established through a BT dongle connected to the USB port. The pairing procedure is available in [131]. The previous tool has been modified to handle BT streams from the wearable devices. Two additional buttons are added to the GUI of the program to start and stop the BT communication.

The device is seen as a serial port where reading and writing bytes. A specific byte is written (sent) to the port to trigger a signal to the wearable device and start the packet transmission. A packet has a length of 22 bytes and the meaning of each field is described in Table 2.3. The first (BOF = 192) and last (EOF = 193) bytes are used to signal the start and the end of the packet. The timestamp, assigned to a generic sample recorded by SHIMMER, is stored in three bytes, while each acceleration value uses two bytes.

Byte No.	1	2	3	4	5	6	7	8	9	10	11
Mean	**BOF**	Sensor ID	**Timestamp**			**Seq. Numb.**	Length	**Acc. X**		**Acc. Y**	
Byte No.	12	13	14	15	16	17	18	19	20	21	22
Mean	**Acc. Z**		Empty		Empty		Empty		**CRC**		**EOF**

Table 2.3.: SHIMMER packet format.

The Sequence number is another useful information to check if a packet is getting lost during the BT transmission. Finally, a CRC is used to check the integrity of the packet. For each sensor, the received packets are saved in a single binary file. A thread is assigned to each sensor in order

to handle separately the BT streams.

An important feature assigned to each packet is the received time at the Pc. The timestamp is calculated using the QPC function and a specific file will be created for each SHIMMER in a dedicated folder. Each entry of the file is the timestamp assigned to the packet.

These programs have been tested in the following platforms:

- Windows 8.1 Pro 64-bit i7-2700K CPU @ 3.50 GHz (8 CPUs), 16 GB RAM;
- Windows 8.1/10 64-bit i5 CPU @ 3.2GHz (4 CPUs), 8 GB RAM.

In all the tests carried out, no frames have been lost during the recording phase, not even when all the streams were enabled.

The datasets recorded with these tools are:

1 **Fall detection dataset**	*Dts_Fall*
Sensors:	Kinect V1
Streams:	depth stream 320×240 @ 30 fps
Device configuration:	the sensor is placed on the ceiling at a height of 3 m from the floor
Number of volunteers:	4 people
Number of tests:	20
Test description:	First group (10 tests): two or more volunteers walk in the monitored area. Second group (10 tests): the volunteer performs some falls in the covered area. He moves some objects and walks very close to other volunteers.
2 **ADL/Fall activities**	*Dts_ADLFall*
Sensors:	Kinect V2 two SHIMMER devices
Streams:	depth frame 512×424 @ 30 fps with timestamps skeleton stream with timestamps two BT streams from the wearable devices with timestamps
Device configuration:	Kinect at 1.5 m from the floor (standard height) and 2 m from the person SHIMMER placed on the belt SHIMMER placed on the left wrist
Number of volunteers:	11 people

Number of tests:	88
Test description:	in different records, each volunteer performs four ADL (sit, grasp, walk and lay) and four falls (front, back, side and end up sitting)
3 **TUG tests**	*Dts_TUG*
Sensors:	Kinect V2 one SHIMMER device
Streams:	depth frame 512×424 @ 30 fps with timestamps skeleton stream with timestamps BT stream from the wearable device with timestamps
Device configuration:	Kinect at 1.5 m from the floor (standard height) and 3 m from the person SHIMMER placed on the chest
Number of volunteers:	20 people
Number of tests:	60
Test description:	the volunteer performs the TUG test, at the beginning he sits on a chair, then he stands up and walks for 3 metres, after that he turns around and goes back to the chair
4 **Food intake dataset V1**	*Dts_intake_1*
Sensors:	Kinect V1
Streams:	depth stream 320×240 @ 30 fps colour stream 640×480 @ 30 fps
Device configuration:	the sensor is placed on the ceiling at a height of 3 m from the floor
Number of volunteers:	35 people
Number of tests:	48
Test description:	the person enters in the monitored area and sits near the table. He starts eating and drinking using glass and cutlery on the table
5 **Food intake dataset V2**	*Dts_intake_2*
Sensors:	Kinect V1
Streams:	depth stream 320×240 @ 30 fps

	IR stream 640×480 @ 30 fps
Device configuration:	the sensor is placed on the ceiling at a height of 3 m from the floor
Number of volunteers:	20 people
Number of tests:	60
Test description:	each person performs three different tests: - eating a snack using the hand and drinking water from the glass; - eating a soup with a spoon and pouring/drinking water; - using knife and fork for the main meal and finally using a napkin.

Chapter 3.

Synchronization Algorithms

This chapter describes the synchronization algorithms used to exploit at the same time the information provided by wearable and camera sensors illustrated in Chapter 2. In particular, the first part analyses the time synchronization between the SHIMMER packets and Kinect streams taking into account eventual drawbacks in BT communications and the delays introduced by Kinect recording process.

The second part concerns the spatial synchronization between the RGB and depth cameras of Kinect devices. Indeed, the raw frames cannot be directly superimposed each other, but a mapping algorithm, that exploits a calibration procedure, is required.

3.1. Kinect and SHIMMER Time Synchronization

Figure 3.1 shows a possible situation where the streams provided by two different sensors have to be synchronized. In details, through the USB 3.0 connection, Kinect V2 sends to the Pc the different streams with a frame rate of approximately 30 fps ($\approx$ 33 ms). Whereas, the wearable device communicates with the Pc through a BT connection and the sampling frequency of the accelerometer sensor is 100 Hz (t_s = 10 ms). Due to the BT connection, a delay must also be included in the total time necessary to send a packet from SHIMMER to the Pc. As introduced in Chapter 2, the communication between SHIMMER and the Pc is bidirectional; a command must be sent to the sensor to start the streaming with the possibility to request the transmission of a single packet. Differently, Kinect streams can only be opened or closed using dedicated APIs. The *Complete Viewer* tools assign time information to each sample received by the Pc.

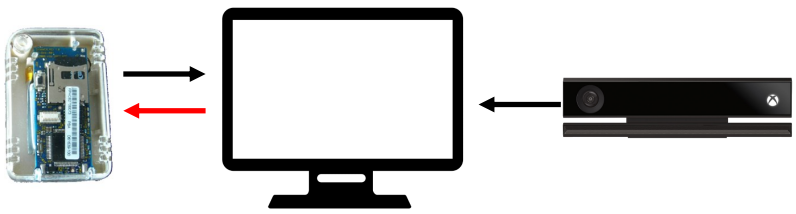


Figure 3.1.: System configuration. The SHIMMER device sends packets to the Pc at 100 Hz while Kinect V2 streams information at approximately 30 fps.

Especially, the timestamps are provided by the C++ functions *QueryPerformanceCounter* (QPC) [129] and *QueryPerformanceFrequency* (QPF) [132]. The QPC provides the number of ticks from the initialization of the operating system. This value is converted in a temporal interval dividing it for the output of the QPF function, i.e. the frequency of the performance counter. The time resolution of the calculated timestamp is close to 1 μs, therefore it is able to handle information sampled with a frequency also higher than 100 Hz. The calculated QPC timestamps are not synchronized to an external time reference, such as the Coordinated Universal Time (UTC), but for this thesis purpose it is enough that the devices are locally synchronized with the Pc.

In Figure 3.2 are visible the three different temporal axes associated to Kinect (t_K), Pc (t_{Pc}) and Shimmer (t_{SH}), respectively. Each one of these axes has a different start time. SHIMMER starts counting time through the resolution of its internal clock oscillator when the device is switched on. Similarly, also the Pc has its clock, not directly related to the previous one. Finally, Kinect clock is not accessible and there is no API providing the time when a frame (depth-IR-RGB) is generated into the device. The frame is sampled at the time t_{on-K}, for an exposure time of t_{EXP} and then sent to the Pc in the time interval t_{TX}. Both t_{EXP} and t_{TX} are not negligible and must be considered for the synchronization algorithm.

On the other hand, SHIMMER samples the acceleration values exactly every t_s. Indeed, considering that the resolution of the internal crystal is 32.768 kHz and only four clock cycles are necessary to sample an acceleration value, the time required to read and convert the acceleration value is only 0.12 ms. Therefore, there are no delays due to the sampling operations. However, the BT communication introduces a variability in the transmission time that is sketched in Figure 3.2 through different time intervals (t_{d1}, t_{d2}).

The synchronization solution, proposed in the next section, allows to estimate the quantities

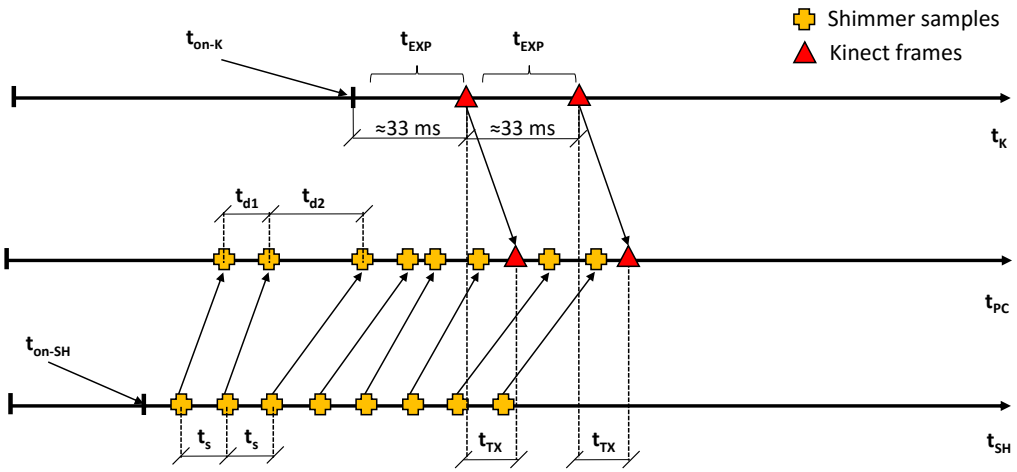


Figure 3.2.: Synchronization issues between Kinect and SHIMMER devices. t_K and t_{SH} axes show the instants when samples are generated, while the received data are represented in t_{Pc} axis.

t_{EXP}, t_{TX} and t_{d1} t_{d2}. In this way, it is possible to find a common reference time and to know when a sample has been generated. Consequently, for each frame will be found the corresponding acceleration sample.

3.1.1. BT Packets Synchronization with the Pc

Figure 3.3a is another representation of the BT communication between the wearable device and the Pc with the related delay issue. In the x-axis there is the time and it started when the Pc has been switched on. The y-axis represents the packets/frames received by the Pc. The red line marks the frames provided by Kinect V2, while the green line refers to the corresponding samples sent by SHIMMER. In this particular case, the wearable device starts to send the packet 2 seconds before the first frame is received by the Pc and ends 1 second later. The line slopes also show the different amount of samples generated by the two devices, in particular for a single frame there are 3 samples provided by the wearable device. It confirms the different sample times, 33 Hz for Kinect and 100 Hz for SHIMMER.

The straight line of the received frames suggests that there are no noticeable variations in the transmission times between consecutive images, but unfortunately this is not true for the packet stream. As stated by Wåhslén in [133], the non-linear trend in the green line of Figure 3.3a is due to the characteristics of the BT communication. The packets, sent by the wearable device, are not received each regular time interval, but sometimes they follow a burst trend. The rectangle in Figure 3.3a, between the green and the red curves highlights this trend. It contains a zoomed portion of the green line and many packets are received to the Pc approximately at the same time.

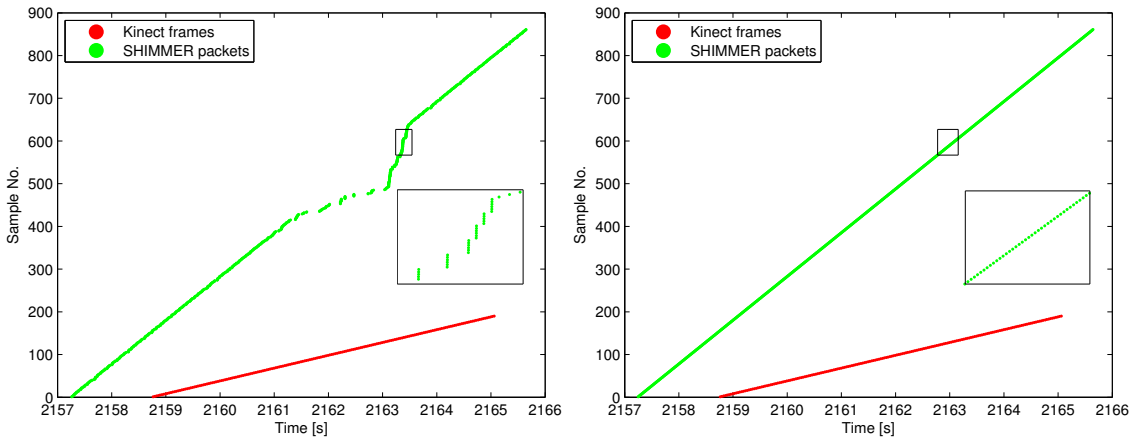
This is a significant problem for an application that implements a data fusion approach, because the streams received by the Pc are not each other related.

Two different solutions can be exploited to solve the problem. In the first one, the burst packets are corrected with a linear regression algorithm. In details, the packets that correctly arrive at the Pc in 10 ms time interval are used to calculate the slope and y-intercept of the green line. Afterwards, the packets in the burst situation are moved to their correct positions using the characteristics of the green line. The result is visible in Figure 3.3b.

This linearization technique is a useful solution when the received time for each packet by the Pc is the only available information. The hypothesis which has to be verified is that all the packets sent from the wearable device are received by the Pc (no lost packets). The condition is respected for all the set-ups used during the tests of the AAL applications described in this work. It has been tested by the analysis of the sequence number field in the received packets (see Table 2.3).

The other approach uses the Marzullo's algorithm [134] and, differently from the previous situation, now it is necessary to know when the packet is generated by the device. The idea is to add an offset (off) to the timestamp (T_{Sen}) assigned by the internal clock of the wearable device to the sent packet. The goal is to have the SHIMMER time axes shifted to the corresponding axes of the Pc.

First of all, it is necessary an initial phase where the system finds the offset. The Pc sends



(a) (b)

Figure 3.3.: Samples at the Pc before (a) and after (b) the synchronization.

a request message to the device at time T_{Pc1}. The wearable device replies to the Pc with a second message containing the time, referred to the t_{SH} axes, when the second packet is generated (T_{Sen}). At the arrival of the second packet, the Pc saves the time of the received reply (T_{Pc2}). This operation is repeated different times and for each iteration, the triplet (T_{Pc1}, T_{Sen}, T_{Pc2}) is saved. The offset is counted as hereunder:

$$off = \frac{min(T_{Pc2} - T_{sen}) + max(T_{Pc1} - T_{sen})}{2} + max(T_{Pc1} - T_{sen}) \tag{3.1}$$

Where ($T_{Pc2} - T_{Sen}$) and ($T_{Pc1} - T_{Sen}$) are calculated for each repetition and the maximum/minimum value are used to find the offset.

Finally, to synchronize the packets with the Pc and solve the burst problem, the timestamp stored in the packet is incremented by the offset calculated in Eq. 3.1.

For the applications proposed in Chapter 5, the first solution has been used because comparing it to the Marzullo's algorithm, it is required only the timestamp assigned by the Pc to the received packet.

3.1.2. Transmission and Exposure Times for Kinect Streams

Contrary to the wireless communication between wearable device and Pc, Kinect uses a cable connection, USB 2.0 for Kinect V1 and USB 3.0 for Kinect V2.

The time interval, starting with the frame acquisition and ending when the image is received by the Pc, can be divided into two contributions. Figure 3.2, shows the exposure time (t_{EXP}) and the transmission time (t_{TX}). In details, t_{EXP} is the time necessary to create the frame, while t_{TX} is the minimum time to send the frame from Kinect to the Pc. Therefore, differently from the BT issues, now there are two variables that must be calculated to find the time when a frame

has been generated. In this case, the Marzullo's algorithm cannot be used, because the internal clock of the device is not accessible and the Kinect stream can only be opened or closed without the possibility to request the transmission of a single frame. At the same time, the linearization allows to align some delayed frames only, but it does not give information about the exposure time. Therefore, other approaches have been implemented to consider t_{EXP} and t_{TX}.

During the tests to estimate these two values, an Arduino Uno V.3 connected to the Pc has been exploited. It drives 7 LEDs and it is positioned in front of Kinect in order to be framed by the camera. A command is sent to the embedded board to switch on and off the LEDs with a known scheme. Evaluating which LEDs are on and off inside the frame captured by Kinect, it is possible to compute the transmission and the exposure times.

The time necessary to send a command to Arduino (1.6 ms) will be discarded in the final formula to estimate t_{EXP} and t_{TX}. It is calculated repeating 1500 transmissions of a single command from the Pc to the board and measuring the time interval between the transmission and the reply.

The scheme in Figure 3.4 illustrates the procedure to estimate the RGB transmission time of Kinect V2. Starting from the top of the figure, the three time axes are respectively associated to the Pc, Kinect and the Arduino board. The sequence of operations are:

- the Pc open the communication with Kinect and waits to receive the data;
- when a frame F0 is received, the Pc saves the arrival time (t_{0_Pc}) and sends the trigger command to the Arduino board to switch on the LEDs;
- the embedded board waits 20 ms before switching on all the LEDs, one every 3 ms;
- when the Pc receives the frame F2, it stores the arrival time (t_{2_Pc}) and the active LEDs in the frame F2 are counted (NUM_LED_ON).

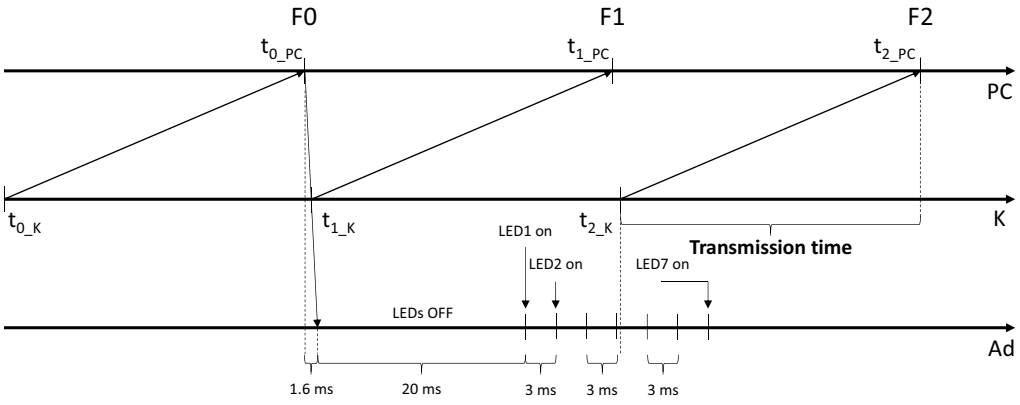


Figure 3.4.: Scheme used to assess the RGB transmission time of Kinect V2.

Eq. (3.2) is used to find t_{2_K} from the information previously described. 1.5 ms is added to consider that, due to a delay of 3 ms between two active LEDs, it is not possible to find the exact instant when the exposure time ends. The delay of 20 ms, found after a test series, allows to have at least a LED off in the frame F2, otherwise there is an ambiguity and the time t_{2_K} cannot be calculated.

$$T_{2\_K} = t_{0\_Pc} + 1.6 + 20 + 3(NUM\_LED\_ON - 1) + 1.5 \tag{3.2}$$

Finally, the transmission time (t_{TX}) is equal to the difference between t_{2_Pc} and t_{2_K}.

For the exposure time, the considered approach is visible in Figure 3.5 and now the considered frames are 4. The sequence of operations are:

- the Pc open the communication with Kinect and waits to receive the stream;
- as in the previous test, when the frame F0 is received, the arrival time (t_{0_Pc}) is saved and a trigger command to the Arduino platform is sent;
- now Arduino works with a different scheme. It switches on all the LEDs, waits for 30 ms and starts switching off the LEDs one by one every 3 ms. In this way, it is possible to find when the exposure time for the frame 3 starts ($t_{3\_K\_start}$).

Considering the LEDs off in the F2 frame, the time $t_{3\_K\_start}$ is:

$$T_{3\_K\_start} = t_{0\_Pc} + 1.6 + 30 + 3(NUM\_LED\_OFF - 1) + 1.5 \tag{3.3}$$

By using the previously computed transmission time, the time t_{3_K} can be found as follows:

$$T_{3\_K} = t_{3\_Pc} - t_{TX} \tag{3.4}$$

The exposure time is equal to the difference between t_{3_K} and $t_{3\_K\_start}$.

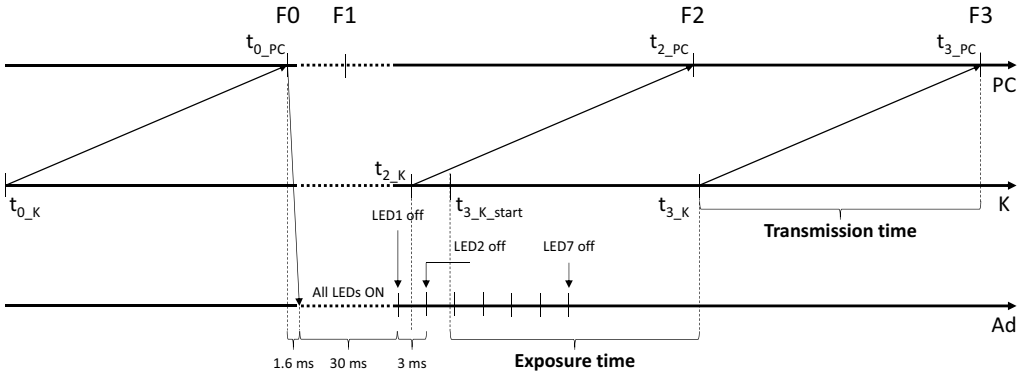


Figure 3.5.: Scheme used to assess the RGB exposure time of Kinect V2.

The following devices have been used to make the experiments:

- Pc desktop with Intel i7 @ 3.5 GHz, 16 GB RAM, Windows 8.1 operating system, for Kinect V2;
- laptop Intel core 2 Duo @ 2.53 GHz, 4 GB RAM, Windows 7 operating system, for Kinect V1.

200 frames have been used for tests with Kinect V1, while 75 with Kinect V2. Visible LEDs have been used for the RGB frames, whereas IR LEDs (830 nm) for the IR and depth frames.

For all the streams provided by Kinect V1 and V2, Table 3.1 shows the mean and the standard deviation of both transmission and exposure times using the techniques previously described.

The exposure time for the RGB-IR-depth frames of Kinect V1 and RGB frame for Kinect V2 are quite similar, while the corresponding values for the IR-depth frames of Kinect V2 are ten times lower. The transmission time for the RGB and IR frames of Kinect V1 is lower than the depth stream, this can be justified with the time necessary to the device to calculate the depth information from the IR frame using the Structured Light principle. On the contrary, the performances of Kinect V2 are inverted. The transmission time of the RGB frame is two times the corresponding value for the Depth-IR frames. This could be related to the size of the RGB frame that is 20 times bigger than the depth frame as well as the technique (TOF) used to calculate the distance is faster than the Structured Light. Finally, taking into account that the skeleton is calculated from the depth image, the time information calculated for that frame can be used for the skeleton too.

Data stream	**Transmission time [ms]**	**Exposure time [ms]**
Kinect V1		
RGB	15.0 ± 2.2	28.4 ± 2.0
IR	16.4 ± 2.0	31.3 ± 2.2
Depth	28.2 ± 5.7	29.1 ± 6.3
Kinect V2		
RGB	31.5 ± 1.1	28.5 ± 1.2
IR	16.0 ± 1.0	3.0 ± 1.2
Depth	18.5 ± 1.0	3.0 ± 1.2

Table 3.1.: Transmission/Exposure time for Kinect V1/V2 with 95% confidence level.

3.1.3. Streams Synchronization

Further to the results of the previous sections, now it is possible to synchronize the information provided by the two sensors. The steps are as follows:

- the SHIMMER packets are linearized;
- the transmission and exposure times of Kinect frame are considered to find when the frame is captured by the device. In particular, the transmission time and half of the exposure time are subtracted from the timestamp of the received frame;
- every SHIMMER packet is associated to the closest frame.

To verify that the synchronization is correct, a specific test is performed. The three LEDs in the SIMMER device are driven by the scheme visible in Figure 3.6. The orange LED is on for 30 ms and only during this interval the packet sent (P_{or}) has a specific value used to separate it from the others. Kinect is in front of the wearable device and records the RGB stream. After the synchronization, the algorithm finds the packets P_{or} using the flag in its field and evaluates the corresponding RGB frame. If the synchronization is correct, the RGB frame shows the orange LED on.

This test has been repeated 48 times for Kinect V2 with a successful percentage of 93.8%. For Kinect V1, the same test is performed 180 times and the successful percentage is 99.4%.

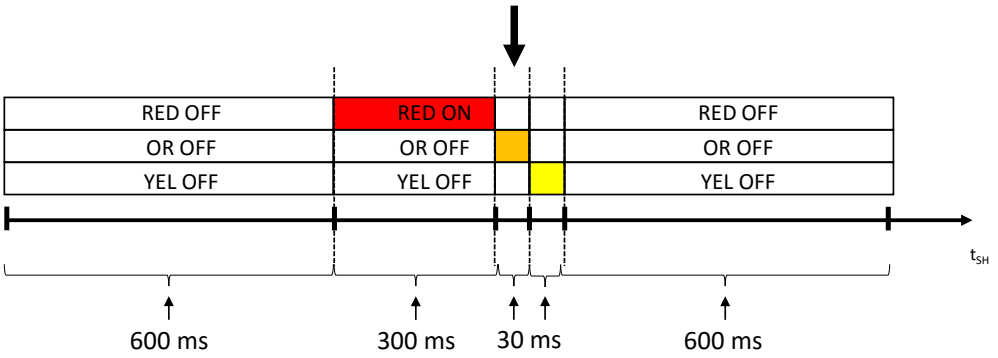
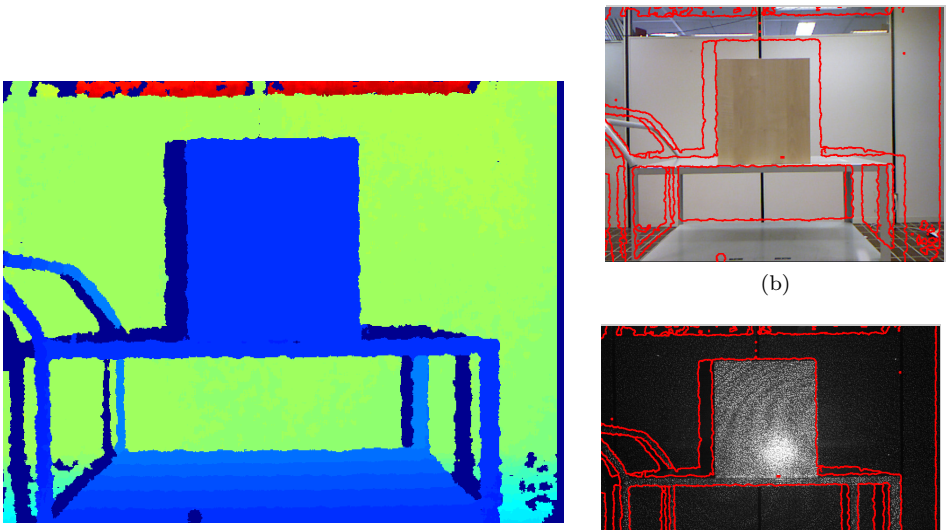


Figure 3.6.: Scheme of the SHIMMER LEDs used to verify the synchronization with Kinect.

3.2. Spatial Synchronization of RGB and Depth Frames

This section treats the mapping problem between the depth and RGB frames. Indeed, due to the different physical location of the two cameras in Kinect, there is not a direct spatial matching between the raw streams in output from the device. In Figure 3.7a is visible a depth frame provided by Kinect V1. A wood square placed on a cart and the different colours are used to represent the distances. The same scene captured by the RGB camera is shown in Figure 3.7b. The Sobel algorithm (A.2) is exploited to find the objects' borders in the depth image. These contours, highlighted in red, have been superimposed to the RGB frame and, obviously, the borders of wood square and cart do not match. This means that a region in the depth frame does not correspond



(a) (b) (c)

Figure 3.7.: Depth frame (a). Depth frame's edges superimposed to RGB (b) and IR (c) frames.

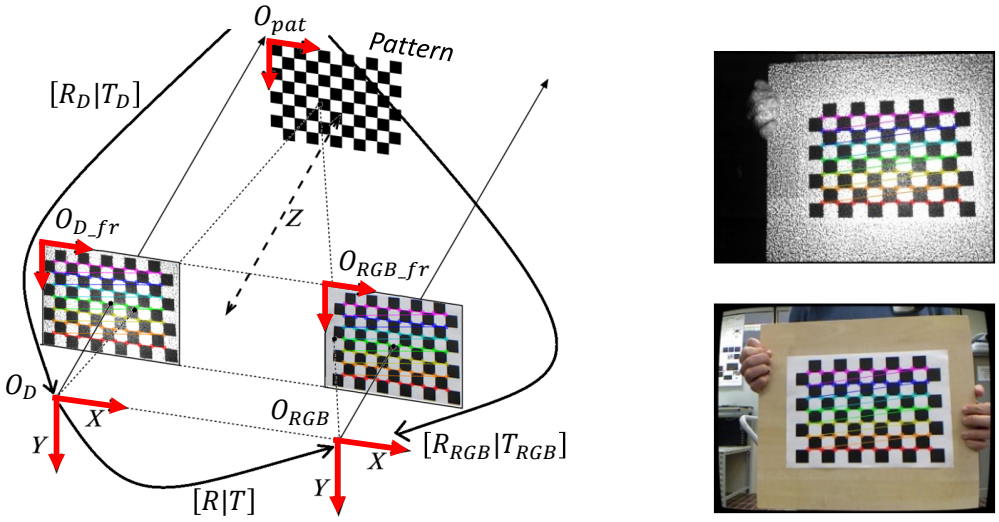
to the same area in the RGB frame. Therefore, a data fusion application, that needs a perfect spatial match, cannot be used directly with the raw frames.

On the other hand, in Figure 3.7c is visible the IR image captured from the same scene. In this case, there is a perfect matching between the depth and IR frames. The reason is that the two images are produced by the same IR sensor. The IR frame is directly provided by the IR camera, whereas the depth frame is calculated from the IR image, as described in Section 2.1.3. Thus, in the following discussion the IR and Depth cameras are related to the same reference system.

3.2.1. Calibration Procedure

The calibration procedure is exploited to find the geometrical parameters of the cameras with the aim to solve the mapping problem previously described. It is necessary to model this process as a stereo calibration problem. The Pinhole model introduced in Section 2.1.1 is now used to represent the RGB and IR cameras. In details, in Figure 3.8a are sketched five reference systems:

- the depth (IR) CCS placed in O_D, centre of projection of the IR camera;
- the RGB CCS placed in O_{RGB}, centre of projection of the RGB camera;
- the depth (IR) FRS centred in O_{D_fr}, the top left corner of the depth frame;
- the RGB FRS centred in O_{RGB_fr}, the top left corner of the RGB frame;
- the pattern coordinate system centred in O_{pat}, the top left corner of the calibration pattern.



(a) (b)

Figure 3.8.: Reference systems for the depth and RGB cameras (a). Chessboard framed by the cameras (b).

The outputs of the calibration algorithm are:

- the intrinsic matrix of the depth (IR) and RGB cameras (K_D, K_{RGB});
- distortion parameters of the depth (IR) and RGB cameras (k_D, k_{RGB});
- the rotation matrix (R) and the translation vector (T) which map the points from the depth CCS to the RGB CCS.

The first four elements ($K_D, K_{RGB}, k_D, k_{RGB}$) have been already introduced in Section 2.1.1, while (R, T) are parameters that link a 3D point in the reference system centred in O_D to the same point, but in the reference system centred in O_{RGB}. The first component is used to rotate the reference system, while the second represents the shift (offset) between the two origins.

$$P_{RGB} = RP_D + T \tag{3.5}$$

In general, (K_D, k_D) and (K_{RGB}, k_{RGB}) can be separately calculated, with two standard camera calibration procedures, whereas a stereo calibration provides (R, T).

A known pattern with features easily identifiable in both the images is used as a calibration object. In particular, as visible in Figure 3.8a, it has a specific combination of white and black squares similar to the grid of a chessboard and the features are the corners of these fundamental elements. In addition, its flat shape reduces the number of unknowns in the equations to solve the calibration procedure. In Figure 3.8b two frames captured by the IR and the RGB cameras are

visible and they have the chosen pattern in foreground. The features are automatically found by the algorithm and are highlighted with different colours.

The idea to exploit this specific geometric pattern is based on the works in [135, 136]. The method proposed by Zhang in [136] is used to find the intrinsic parameters, whereas the distortion coefficients are calculated with the Brown's solution [137]. In particular, the calibration object is shown to the cameras with different angles, at least for 10 times. Each point $P = [x_{pat}\ y_{pat}\ z_{pat}]^T$ (corner) in the pattern reference system could be related to the CCSs thanks to a combination of a rotation matrix and a translation vector. For example, for the depth camera:

$$\lambda \begin{bmatrix} u_D \\ v_D \\ 1 \end{bmatrix} = K_D \begin{bmatrix} R_D & T_D \end{bmatrix} \begin{bmatrix} x_{pat} \\ y_{pat} \\ z_{pat} \\ 1 \end{bmatrix} \tag{3.6}$$

$$R_D = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad T_D = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \quad K_D = \begin{bmatrix} f_{D\_x} & 0 & c_{D_x} \\ 0 & f_{D\_y} & c_{D_y} \\ 0 & 0 & 1 \end{bmatrix}$$

where the points are in homogeneous coordinates format and λ is a constant. This relation, setting $z_{pat} = 0$ with no loss of generality, is called *planar homography*, because maps a point from one plane (chessboard) to another (frame).

For every specific image framing the object in different poses, there is a related equation equal to Eq. (3.6) where R_D and T_D change and K_D is always the same. Solving the system of equations it is possible to estimate the intrinsic parameter of the depth camera. A similar consideration is possible also for the RGB camera replacing $(u_D v_D K_D R_D T_D)$ with $(u_{RGB} v_{RGB} K_{RGB} R_{RGB} T_{RGB})$.

In Eq. (3.6) the frame coordinates refer to a depth frame, but in practice the calibration procedure uses the IR frame as it is impossible to find the chessboard's corners in the depth frame, since they are flat elements. As explained in Figure 3.8b, depth and IR frames are perfectly overlapped, thus K_D is the intrinsic parameter of the IR camera.

To estimate the distortion coefficients of the camera, another system of equations must be solved. It is formed by Eq. (2.5) where the unknowns are the distortion coefficients.

It is now possible calculating the (R, T) from (R_D, T_D) and (R_{RGB}, T_{RGB}) for each object pose (stereo calibration). The same 3D point P, in the pattern reference system, is mapped in the depth/RGB CCS:

$$P_D = R_D P + T_D \qquad P_{RGB} = R_{RGB} P + T_{RGB} \tag{3.7}$$

Combining the two equations and writing P_{RGB} in function of P_D, the extrinsic (or external) parameters, to map a point from the depth CCS to the RGB one, are:

$$R = R_{RGB}(R_D)^T \qquad T = T_{RGB} - RT_D \tag{3.8}$$

The R and T found for each pose of the object, due to image noise, are slightly different each other, but using a minimization error algorithm the best (R, T) are calculated.

Now all the intrinsic and extrinsic parameters are found and therefore the steps to map a point $([u_D \ v_D]^T)$ of a depth frame in the RGB image are:

- use the distortion parameters of the depth camera with Eq. (2.5) to find the corrected depth frame;
- the same correction formula is used in the raw RGB frame, replacing the distortion parameter k_D with k_{RGB};
- reverse Eq. (2.4) to find the corresponding 3D depth point $([x_D \ y_D \ z_D]^T)$ in the reference system of the depth camera (centred in O_D).

$$\begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} = \lambda K_D^{-1} \begin{bmatrix} u_D \\ v_D \\ 1 \end{bmatrix} \tag{3.9}$$

- use the extrinsic parameters to find the equivalent point in the RGB CCS:

$$\begin{bmatrix} x_{D\_map} \\ y_{D\_map} \\ z_{D\_map} \end{bmatrix} = R \begin{bmatrix} x_D \\ y_D \\ z_D \end{bmatrix} + T \tag{3.10}$$

- exploit Eq. (2.4) to project the 3D point from the RGB camera system (centred in O_{RGB}) in the RGB frame system (centred in O_{RGB_fr}).

$$\lambda \begin{bmatrix} u_{D\_map} \\ v_{D\_map} \\ 1 \end{bmatrix} = K_{RGB} \begin{bmatrix} x_{D\_map} \\ y_{D\_map} \\ z_{D\_map} \end{bmatrix} \tag{3.11}$$

At the end of these operations, the depth pixel $([u_D \ v_D]^T)$ is mapped in the RGB frame $([u_{D\_map} \ v_{D\_map}]^T)$.

3.2.2. Performances Evaluation

Different alternative solutions are available to map the depth frame in the RGB image. For example, the official Microsoft SDK (1.5-1.8) and the OpenNI SDK 2.1 library provide specific functions for this purpose:

- *NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution*: it a function in the SDK 1.5 [138] that gets a single pixel per time;

– *NuiImageGetColorPixelCoordinateFrameFromDepthPixelFrameAtResolution*: it is a function in the SDK 1.6 [139] or higher that works with more pixels per time;

– *convertDepthToColor*: it is a function of the class *CoordinateConverter* in OpenNI SDK 2.1 [140] that gets a single pixel per time.

The official development groups, who wrote these libraries, did not disclose a documentation that explains the mapping solution implemented in the previous functions. These methods are "black box" functions and it is not necessary a calibration procedure to find the camera parameters, but just the depth point to map must be passed to the function to get the corresponding RGB point in output.

RGBDemo [141] is another tool used to handle Kinect streams and compute the camera parameters. It needs a calibration procedure that exploits the OpenCV functions [93, pp. 415–453]. Differently from the RGBDemo solution, the technique described in this section takes also into account the intrinsic parameters of the depth camera during the mapping process.

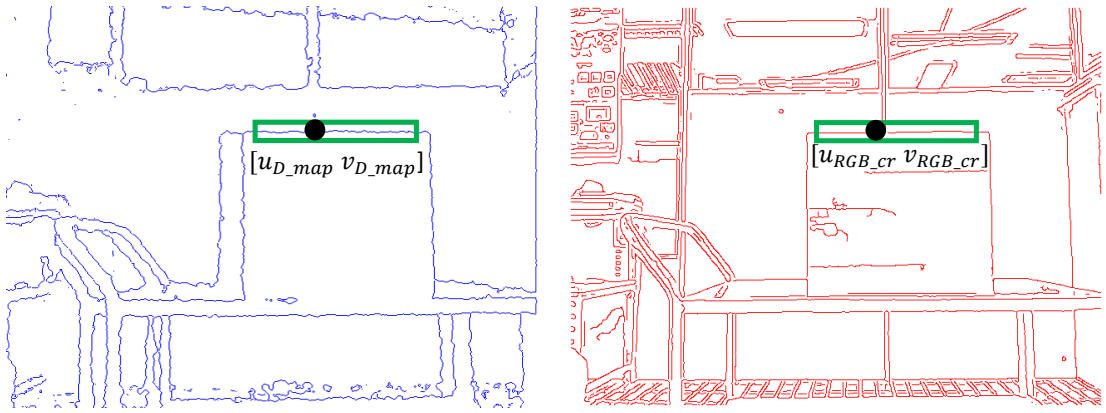
Different tests have been performed to estimate the mapping accuracy of the previous solutions. One idea is to calculate the borders in both the RGB and mapped depth frames. Then, the cross-correlation could be used to find the mapping algorithm that has less misalignments. As visible in Figure 3.7, unfortunately there are some artefacts, not present in the RGB frame, which can influence the cross-correlation result. These components are the zero pixel areas in the bottom right region of the frame and the shadows in the depth image.

The simplest solution is to limit the analysis to the borders of a regular object, such as the wood square in Figure 3.7. Its dimensions are 39.6×35.1×1 cm and it is easily identifiable in both the depth and the RGB frame.

It is placed at three different distances from the sensor (1-2-3 m), in the centre of the frames. The same tests are repeated with the target placed in a side of the image. Kinect V1 and the target are respectively placed at 92 cm and 87.4 cm from the ground. The resolution of the frames used in the tests is 640×480 pixels. The borders of the object are calculated in the mapped depth frame using the Sobel algorithm and the same operation is repeated in the RGB frame with the Canny algorithm.

Each test evaluates the vertical mismatch in pixel coordinates for the top border of the object and also the horizontal mismatch for the right border. As visible in Figure 3.9, $[u_{D\_map}\, v_{D\_map}]^T$ is a generic pixel border after the mapping and $[u_{RGB\_cr}\, v_{RGB\_cr}]^T$ is a point in the undistorted RGB frame.

$$\begin{aligned} err_{top} &= \frac{1}{v_{max} - v_{min}} \sum_{v=v_{min}}^{v_{max}} |\overline{u}_{D\_map} - \overline{u}_{RGB\_cr}| \\ err_{side} &= \frac{1}{u_{max} - u_{min}} \sum_{u=u_{min}}^{u_{max}} |\overline{v}_{D\_map} - \overline{v}_{RGB\_cr}| \end{aligned} \tag{3.12}$$



(a) (b)

Figure 3.9.: A portion of the mapped depth frame of the target at 1 m from the cameras (a). Portion of the distortion-free RGB frame of the same scene (b).

For the top border, a horizontal range ($[v_{min}, v_{max}]$) is manually set equal to the length of the target's top edge. In Figure 3.9 the horizontal range is highlighted with a green rectangle that covers the top border. For each pixel of the interval is identified a vertical index of the border in the depth ($\overline{u}_{D\_map}$) and RGB ($\overline{u}_{RGB\_cr}$), the difference between these values is the mismatch. The mean value (err_{top} [pixel]) represents a quality parameter to compare the different mapping solutions. The same error is calculated for the right border using the second formula in Eq. (3.12). The green rectangle is now placed in the right border ($[u_{min}, u_{max}]$) and the mismatching is calculated between column coordinates.

Table 3.2 shows the mean errors for both the top and right edge, when the target is placed at 1 m, 2 m and 3 m in the centre of the frame. SDK 1.5 refers to the function *NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution* and SDK 1.6/1.8 is used for *NuiImageGetColorPixelCoordinateFrameFromDepthPixelFrameAtResolution*. For each distance, the values in bold represent the minimum errors, whereas the maximum values are underlined. The Average columns store the mean errors for three distances and edges. Table 3.3 has the same entries, but now the target is in the right side of the frame.

In most of the cases the proposed solution reaches the lowest error. If this is not verified, its maximum mismatching is below 2.3 pixel. Sometimes, for the other algorithms, the mapping error is quite high. In particular, the maximum errors are:

- SDK 1.5: 11.07
- SDK 1.6/1.8: 7.69
- OpenNI: 9.05
- RGBDemo: 3.16

When the columns of the average values are considered, the proposed solution outperforms the other algorithms three times out of four. Only for the top edge in central situation the SDK 1.6/1.8

mapping solution is better than the proposed, but the difference is less than 1 pixel.

	Top edge				**Right edge**			
	1 m	2 m	3 m	Average	1 m	2 m	3 m	Average
SDK 1.5	**1.08**	**0.54**	3.16	1.59	2.39	9.68	11.07	7.71
SDK 1.6/1.8	1.12	0.810	1.30	**1.08**	4.89	2.04	4.07	3.67
OpenNI	1.12	0.740	3.07	1.64	1.09	7.94	9.05	6.03
RGBDemo	1.95	1.14	1.93	1.67	**0.670**	1.31	0.420	0.8
Proposed	1.58	1.24	**1.23**	1.35	0.960	**0.710**	**0.660**	**0.780**

Table 3.2.: Mapping error [pixel] in top and right central configuration.

	Top edge				**Right edge**			
	1 m	2 m	3 m	Average	1 m	2 m	3 m	Average
SDK 1.5	0.890	1.04	2.65	1.53	1.75	6.90	8.49	5.71
SDK 1.6/1.8	1.37	1.31	2.85	1.84	7.69	**0.840**	**1.17**	3.23
OpenNI	0.870	0.960	2.68	1.50	2.56	4.77	6.45	4.59
RGBDemo	1.16	0.820	**0.630**	0.870	2.44	1.86	3.16	2.49
Proposed	**0.860**	**0.710**	0.830	**0.800**	**1.16**	1.19	2.28	**1.54**

Table 3.3.: Mapping error [pixel] in top and right side configuration.

Chapter 4.

Self-Organizing Networks

A neural network is a system used to model the way in which the brain executes specific tasks. It is accomplished by the collaboration of fundamental computing elements, called neurons, connected each other by a huge number of links (synapses). The brain has the ability to modify its connections or, more in general, its structure in order to generate "knowledge", i.e. be more adaptable to the surrounding environment. *Learning* capability is the process that allows these changes. For example, starting from birth the human brain is continuously stimulated by interactions with the environment. The senses encode this information in electrical signals that are a suitable format to be processed by the brain.

In the same way, in an artificial neural network, two different strategies can be implemented to stimulate the system. The first one is called *learning with a teacher* (supervised learning), where a group of labelled examples are shown to the network. It means that, for each input the system receives also the desired response that it should provide in output. The learning process will modify the network configuration in order to reduce the difference between the desired response and the real response of the system using a specific minimization solution.

Differently, in the *learning without a teacher* (unsupervised learning) the inputs (stimuli) are not labelled and the network itself, through a set of competitive-learning rules, will find the best configuration in order to be more similar to the input. When these learning rules tend to follow the neurological structures of the human brain, the process is called self-organization. The fundamental principles of the self-organization are the following [142]:

- *self-amplification*: the neuropsychologist D. O. Hebb has postulated this principle in 1949, in the context of the human learning process. It explains the principle called *synaptic plasticity*, where the strength in a synaptic increases if a neuron is repeatedly and constantly fired by another neuron. The result is an improvement in communication between the two cells;

- *competition*: due to the limited resources, each neuron is in competition with the others. Inducing a grow in strength for synapses that are fired most of the time and, on the other hand, a decrease for the less used cells that can eventually lead to deleting the connection. The neuron that wins the competition is called the Best Matching Unit (BMU) and, the competition phase is usually followed by a cooperation phase;

- *cooperation*: when a neuron is fired, usually it excites also the neurons that are close to it.

In a system that implements the principle of the self-organization, this process is carried out with a small modification in the network around the BMU;

– *structural Information*: there must be a correlation between the stimuli. For example, in a speech or in a video sequence, when the sampling frequency is adequate, there is a higher correlation between samples. It follows at once that there is a redundancy in the input data. This is a fundamental prerequisite of the self-organized learning and if the property is not verified, the unsupervised learning cannot be exploited.

In this chapter are introduced three different unsupervised neural networks based on principles of the self-organization: Self-Organizing Map, Extended SOM and Growing Neural Gas networks. They will be exploited in the intake monitoring application described in Section 5.3 to track movements performed by a person.

Before the description of the algorithms, the definitions of the fundamental elements of a self-organizing network are given:

– *output space*: it is a discrete space where the neurons are located, connected by edges;
– *neurons/vertices*: it is the fundamental element of the network in the output space. The elements are connected each other using edge/path. No values are assigned to paths (unweighted connections), because they are used only to represent the network topology;
– *maximum number of neurons* (N): it is the number of elements in the network;
– *input space*: it is the space where the input data to the algorithm are placed;
– *manifold* (M): it is a subgroup of the input space that coincides with all the elements in input to the algorithm;
– *weight vectors* (w)*/reference vector*: it is a position in the input space related to a specific neuron. As visible in Figure 4.1, each neuron in the output space references a weight vector in the input space and the complete structure is called model or graph;
– *input signal/stimulus* (ξ): for each element of the manifold, a stimulus is a possible input signal to the self-organizing algorithm (i.e. one grey dot in Figure 4.1).

4.1. Self-Organizing Map

The Self-Organizing Map was proposed by Kohonen [143] in 1982 as clustering solution neurobiologically inspired. It implements the fundamental elements of the self-organization and mimics the learning phase in the human brain. In particular, the neural processes in the central nervous system map the sensor experiences (visual, tactile, etc.) in the cerebral cortex in an orderly fashion. The aim is to make this information promptly available to the higher process of the brain, thanks to the optimized connection scheme between the neurons.

In the same way, using a competitive-learning process, the SOM is able to correlate similar inputs to elements in the network that are close together, whereas different stimuli are mapped in

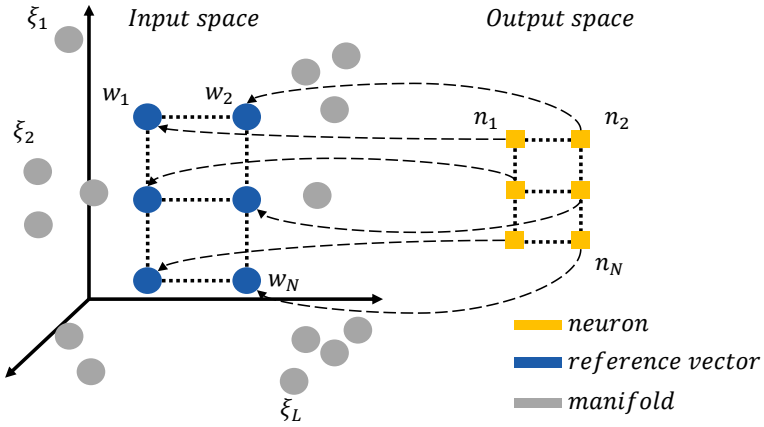


Figure 4.1.: Fundamental elements in a self-organizing network.

nodes that are far away. The first part of the name "self-organizing" comes from the group of the algorithms whose the SOM belongs, i.e. the unsupervised machine learning solutions. The word "map" refers to the properties of linking the input data to the reference vectors. The algorithm, due to its general definition, is used in very heterogeneous situations like industrial analyses, biomedical studies and financial applications, to just mention a few of them. A survey of papers that use the SOM is presented in [144].

As similar to the K-means clustering algorithm, here the mapping is many-to-one. In particular, the input data is averaged by the model in such a way that the distribution of the weight vectors is similar to the distribution of the input. At the same time, the advantage is in the reduced number of network's elements that is a small percentage of the dimension of the input data. The principal difference with the K-means solution is that the SOM also takes into consideration the topographic relations between the reference vectors and the input.

4.1.1. Algorithm Description

Now the fundamental parts of the SOM will be described. As previously introduced with the principles of the self-organization, the SOM is a stepwise-recursive learning algorithm where the most important steps are: initialization of the network, competition, cooperation and reference vector adaptation. In each step, the algorithm tunes some parts of the network using the input signal.

4.1.1.1. Initialization of the Network

The most common configuration of the model is an array or lattice and, in particular, the structure is illustrated in Figure 4.1 in the output space. When the algorithm is used for the first time, the position of each reference vector is randomly set in the input space or alternatively they can take the position of some stimuli that will be shown later to the algorithm. Another strategy, not

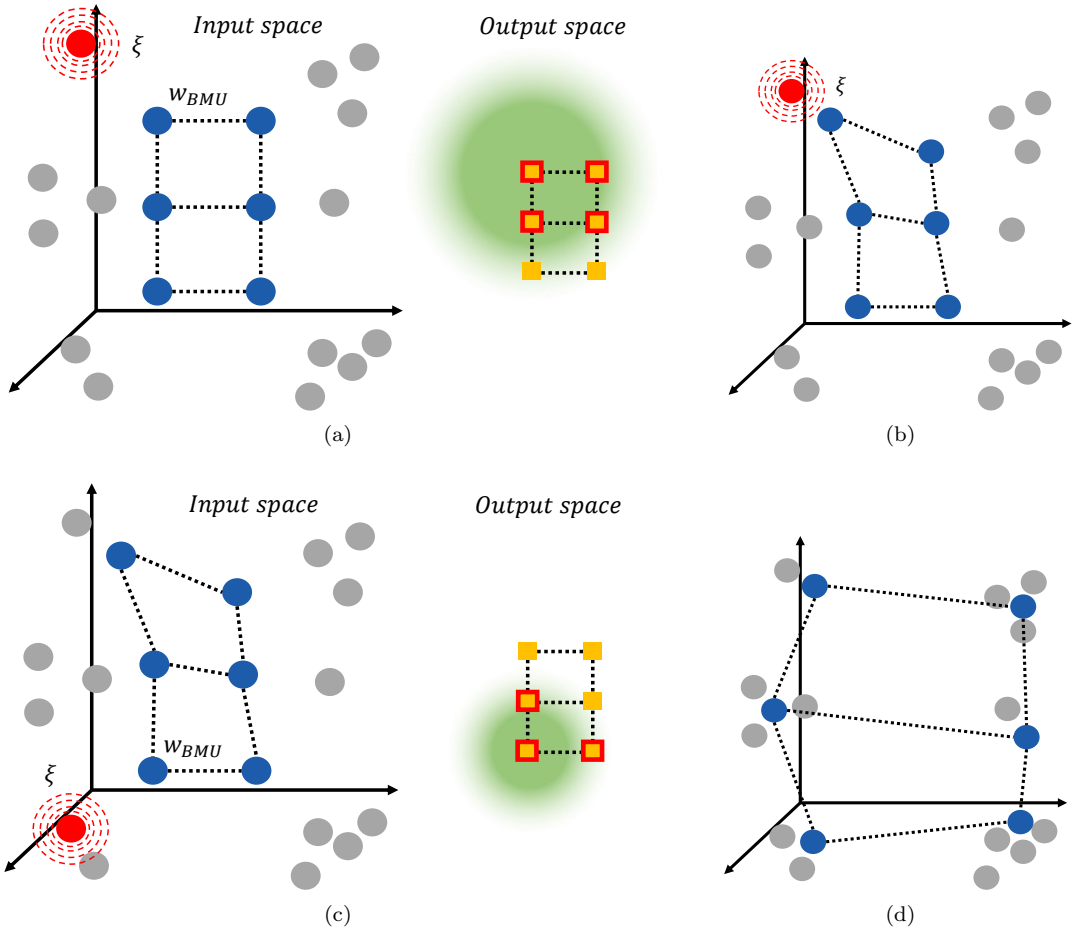


Figure 4.2.: SOM algorithm steps. First input vector to the algorithm, identification of the BMU and neighbouring function (a), adaptive process (b). Second input vector to the algorithm, neighbouring function (c) and final network (d).

covered in the original algorithm, is to use a network that takes into account the configuration of the input space in order to speed up the adaptation process. In Figure 4.2a the reference vectors are placed in the centre of the manifold.

4.1.1.2. Competition

At each step (p) a generic vector, represented by the red dot in Figure 4.2a, is given in input to the algorithm. It is defined as:

$$\xi = [\xi_x \ \xi_y \ \xi_z \ ...] \qquad \{\xi_1 \ \xi_2 \ ... \ \xi_L\} \in M \quad p = 1, 2, 3, ..., L \tag{4.1}$$

The total number of stimuli that compose the manifold in input to the SOM is equal to L. The dimensionality of ξ is not constrained a priory by the SOM, but it should be the same of the reference vector. For example in Figure 4.1, the dimensionality is equal to 3 ($\xi = [\xi_x \ \xi_y \ \xi_z]$). The reference vector (w) of the j^{th} neuron is denoted by:

$$w_j = [w_{jx} \ w_{jy} \ w_{jz} \ ...] \qquad j = 1, 2, 3, ..., N \tag{4.2}$$

The total number of weight vectors is set before using the SOM and it is equal to N.

The SOM finds the BMU as the reference vector that minimizes the Euclidean distance in the input space between ξ and w. In Figure 4.2a the node is labelled with w_{BMU}.

$$w_{BMU} : \|\xi - w_{BMU}\|^2 \leq \|\xi - w_j\|^2 \qquad j = 1, 2, 3, ..., N \tag{4.3}$$

In particular, the algorithm finds the square of the distance because it is not necessary to calculate the specific separation length but only ranking the nodes from the closer to the further.

4.1.1.3. Cooperation

The BMU represents the centre of the adaptation, but it is not the only element that has to be moved. In addition, its neighbours must be updated (excited nodes) in the interest of the cooperation rule introduced in the self-organized learning. In particular, the heuristic function $h_{BMU,j}$ is called *neighbourhood function* and it is used to select the nodes, close to the BMU, that participate to the adaptive process:

$$h_{BMU,j}(p) = e^{d^2_{BMU,j}/2\sigma(p)^2} \qquad \sigma(p) = \sigma_0 e^{-p/\tau 1} \tag{4.4}$$

$h_{BMU,j}$ is a Gaussian function, selected for its translation invariant property, with the maximum centred in the BMU. It means that the function is independent by the position of the BMU. As stated by Kohonen, the function resembles the kernel usually used in smoothing processes [145].

$d^2_{BMU,j}$ is the square distance, calculated in the discrete lattice space of the neurons, between the BMU and the j^{th} node in the network. The parameter is used to find the closest neighbours of the BMU inside the output space. In Figure 4.2a $h_{BMU,j}$ is depicted with the green filled circle

where deep colour is used to represent regions where the function has high values, while moving toward the external areas its intensity decreases until no nodes are involved in the update process.

An important feature of the SOM is the dependence of $h_{BMU,j}$ to the variable p, i.e. the index of the input signal to the algorithm. Therefore, the neighbourhood function shrinks in time and in the final part of the process only the close nodes to the BMU participate to the adaptation. This feature is visible when the shapes of the $h_{BMU,j}$ in Figure 4.2a and Figure 4.2c are compared. In the second case, the area of the green circle is reduced. Obviously, this representation has been implemented only for description purpose. In the real case, between two consecutive ξ, the difference in $h_{BMU,j}$ functions is not noticeable.

As indicated in Eq. (4.4), a reasonable choice for the σ parameter is an exponential decay with σ_0 and $\tau 1$ equal to constants. In particular, σ_0 can be set equal to the lattice's radius, i.e. the distance that covers all the nodes in the output space.

Another choice for the neighbourhood function is called "bubble" form [146], where $h_{BMU,j}$ is not null (equal to 1) only for a specific radius around the BMU.

4.1.1.4. Reference Vectors Adaptation

The last step to perform in each iteration is the adaptation of the nodes to the stimulus. The self-organized learning requires that the reference vector of the winner neuron and its neighbours change their positions in relation to the input signal. The result will be a network where adjacent neurons in the lattice will have similar reference vectors in the input space. The question is: how should this modification be implemented?

The update must be taken in the direction that leads to a modified model that will approximate better the input space. Therefore, the equation that implements this idea is:

$$\Delta w_j = \varepsilon(p) h_{BMU,j}(p)(\xi - w_j) \qquad j : indexes\ of\ active\ nodes \tag{4.5}$$

w_j represents the reference vectors activated by the neighbourhood function and Δw_j is the displacement to be calculated. Reference vectors of the neurons that have a neighbourhood function close to 0 will not be updated. Looking at Figure 4.2a-b, the nodes involved in the update are the elements inside the green circle. In particular, the BMU and the other three reference vectors close to it. Each element is moved toward the direction of the stimulus and the displacement $(\xi - w_j)$ is weighted by the neighbourhood function and $\varepsilon(p)$, i.e. the *learning rate function*. It is monotonic decreasing scalar term, whose trend can be chosen from hyperbolic, piecewise linear or exponential function. A possible example of $\varepsilon(p)$ is:

$$\varepsilon(p) = \varepsilon_0 e^{-p/\tau 2} \tag{4.6}$$

where $\tau 2$ and ε_0 are two constant parameters that must be identified in the tuning phase of the algorithm, taking into account the characteristics of the input data and the time constraint for the convergence of the model. Finally, it is worth noting that the $\varepsilon(p)$ should never reach the 0, otherwise the system will be in an unstable state (*metastable state*) because, in this situation, no

one reference vector will be updated with Eq. (4.5).

Figure 4.2c shows the elements that will be modified when the next input vector is used as input to the SOM. With respect to the previous ξ, here the reference vectors to update are three instead of four. It is due to the neighbourhood function that now covers less nodes.

When the SOM complete the analysis of the manifold after thousands of iterations, the final network is visible in Figure 4.2d. The lattice is adapted and, if the SOM continues the execution, no further noticeable differences will be found.

4.2. Extended SOM

Coleca et al. have recently proposed an alternative [147, 148] to the algorithm previously described. This solution, called Extended SOM, tries to overcome the limits of a previous work that uses the SOM algorithm [149]. The most important innovation is the introduction of new fundamental elements in the structure of the network. Indeed, in the original SOM, the nodes have been used as the only basic components of the graph, but this feature could lead to an unstable model if the input to the SOM is a 3D data such as a PC.

4.2.1. Algorithm Description

The fundamental elements in the networks are segments and planes. In the first case, a segment is made up of two connected nodes, whereas in the other case, three points form a plane.

To fit the network to the input, as well as for the SOM, there is a competition phase where the closest element to ξ is identified. There are three possible alternatives, indeed in the SOM_Ex the BMU can be a single node, a segment or a plane.

For each one of these elements, the corresponding distance (d) from the input must be calculated. When the BMU is a single point, there is no difference with the standard SOM. The closest node to ξ is found using the Euclidean distance.

For the segment and the plane, the projection point (prj) of ξ on these components is calculated. In Figure 4.3 the two situations are illustrated and the point prj is highlighted in magenta colour. As visible, in both the cases, prj compared to the vertices (w) is the closest point to ξ. For the segment, the vectors directed from w_i to w_j and from w_j to w_i are defined:

$$s_{ji} = w_j - w_i,\ \hat{s}_{ji} = \frac{s_{ji}}{\|s_{ji}\|} \qquad s_{ij} = w_i - w_j,\ \hat{s}_{ij} = \frac{s_{ij}}{\|s_{ij}\|} \tag{4.7}$$

where s_{ji} is the vector from w_i to w_j and s_{ji} is the vector in the opposite direction. The unitary vectors related to the previous are $\hat{s}_{ji}$ and $\hat{s}_{ij}$.

The projection point prj is obtained from Eq. (4.8).

$$prj = w_i + n_{ji}s_{ji}\ \ 0 \leq n_{ji} \leq 1 \qquad prj = w_j + n_{ij}s_{ij}\ \ 0 \leq n_{ij} \leq 1 \tag{4.8}$$

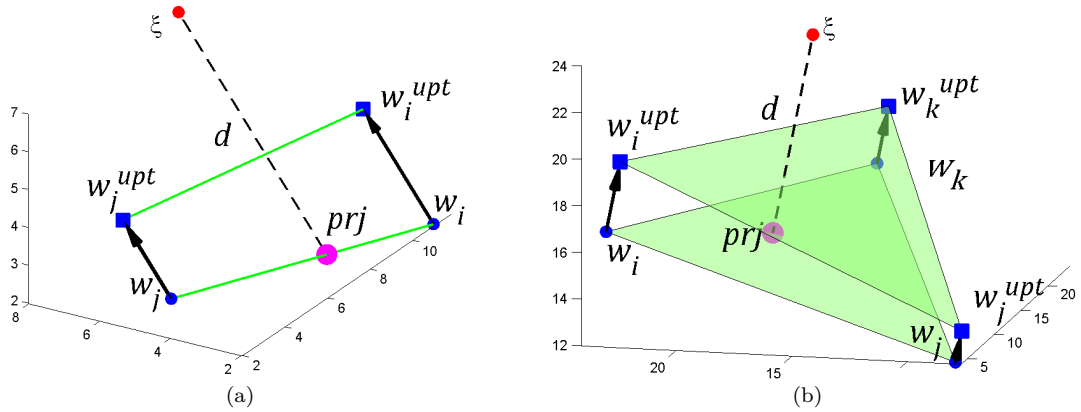


Figure 4.3.: Position updating for a segment (a) and plane (b), according to the input.

where:

$$n_{ij} = \frac{(\xi - w_j)^T \hat{s}_{ij}}{\|s_{ij}\|} \qquad n_{ji} = \frac{(\xi - w_i)^T \hat{s}_{ji}}{\|s_{ji}\|} \tag{4.9}$$

The distance d between prj and ξ is equal to:

$$d = d_{ij} = d_{ji} = \|\xi - prj\| \tag{4.10}$$

It is worth noting that the distance d is valid only if the inequalities in Eq. (4.8) are verified. In the other case, it means that prj is outside the segment and it has not sense find the distance d_{ij} (or d_{ji}) because the closest point to ξ is now one of the vertices.

For the plane, the relevant vectors are:

$$s_{ji} = w_j - w_i \qquad s_{ij} = w_i - w_j \qquad s_{ki} = w_k - w_i \tag{4.11}$$

The projection point is calculated from Eq. (4.12).

$$\begin{aligned} prj &= w_i + n_{ji}s_{ji} + n_{ki}s_{ki} & 0 \le n_{ji}, \quad n_{ki} \le 1, \quad n_{ji} + n_{ki} \le 1 \\ prj &= w_j + n_{ij}s_{ij} + n_{kj}s_{kj} & 0 \le n_{ij}, \quad n_{kj} \le 1, \quad n_{ij} + n_{kj} \le 1 \\ prj &= w_k + n_{ik}s_{ik} + n_{jk}s_{jk} & 0 \le n_{ik}, \quad n_{jk} \le 1, \quad n_{ik} + n_{jk} \le 1 \end{aligned} \tag{4.12}$$

where:

$$\begin{aligned} n_{ji} &= \frac{(\xi - w_i)^T \hat{s}_{ji} - ((\xi - w_i)^T \hat{s}_{ki})(\hat{s}_{ki}^T \hat{s}_{ji})}{\|s_{ji}\|(1 - (\hat{s}_{ki}^T \hat{s}_{ji})^2)} \\ n_{ki} &= \frac{(\xi - w_i)^T \hat{s}_{ki} - ((\xi - w_i)^T \hat{s}_{ji})(\hat{s}_{ki}^T \hat{s}_{ji})}{\|s_{ki}\|(1 - (\hat{s}_{ki}^T \hat{s}_{ji})^2)} \end{aligned} \tag{4.13}$$

For the plane, there are only two degrees of freedom, therefore the other four coefficients n_{ij}, n_{ik}, n_{jk} and n_{kj} are related to Eq. (4.13) using these equalities:

$$n_{jk} + n_{ji} + n_{kj} + n_{ki} + n_{ik} + n_{ij} = 2 \qquad \begin{aligned} n_{jk} &= n_{ji} \\ n_{kj} &= n_{ki} \\ n_{ik} &= n_{ij} \end{aligned} \tag{4.14}$$

The distance between the plane and the input could be calculated from Eq. (4.10) again. In addition, as well as for the segment, the formulas continue to be valid only if inequalities in Eq. (4.12) are verified.

In the competition phase, the different distances d are calculated for each component (node, segment, plane) that belongs to the network. The BMU will be the element with the lowest value of d.

Finally, in the adaptation phase, the position of the BMU must be updated considering the distance d. The equation to use is related to the specific element to be moved. For a single point, the update in Eq. (4.15) is similar to the formula used for the standard SOM (see Eq. (4.5)), where $\varepsilon(p)$ is the learning rate function.

$$w^{upt} = w + \varepsilon(p)(\xi - w) \tag{4.15}$$

When the BMU is a segment, the shift must be done in the direction that decreases the distance d. To this aim, the gradient-descent minimization is used on the square distance to calculate the displacement. It leads to the following equations:

$$\begin{aligned} w_i^{upt} &= w_i + \varepsilon(p) n_{ij}(\xi - prj) \\ w_j^{upt} &= w_j + \varepsilon(p) n_{ji}(\xi - prj) \end{aligned} \tag{4.16}$$

For the plane, the updated formulas are very similar to the previous ones:

$$\begin{aligned} w_i^{upt} &= w_i + \varepsilon(p) n_{ij}(\xi - prj) \\ w_j^{upt} &= w_j + \varepsilon(p) n_{ji}(\xi - prj) \\ w_k^{upt} &= w_k + \varepsilon(p) n_{ki}(\xi - prj) \end{aligned} \tag{4.17}$$

As visible in Figure 4.3a, after a single adaptation phase, the segment is attracted by the input.

In particular, the point w_i compared to w_j is the closer point to ξ and for this reason, its shift is greater than the displacement of w_j. Using Eq. (4.17) in the plane situation, the result is shown in Figure 4.3b.

4.3. Growing Neural Gas

The third self-organizing network is called "Growing Neural Gas". The attribute "growing gas" comes from the behaviour of the nodes in the network that reminds the expanding properties of a gas in a closed space. It has been proposed by Fritzke in [150] and it is based on the work of Martinetz et al. in [151] with the Neural Gas network (NG). The GNG, compared to the NG, is able to add or delete nodes and it uses a different adaptation scheme. From the topology point of view, the NG reaches a better accuracy than the GNG, but the required computational time is greater [152].

The GNG can be exploited to solve two important issues. The *clustering*, when it is necessary to reduce the density in the input dataset. On the other hand, to provide a model of the input that preserves the topology (*topographic mapping*) using a reduced number of reference points. In this work, the GNG is exploited for this second property.

The most important differences between GNG and SOM are:

- in the GNG a pre-defined model is not required, but it is requested to set the position of two nodes only. It is useful when there is not enough information of the input space. Whereas, in the SOM the start model must be chosen similar to the manifold;
- the learning coefficients are constant, while in the SOM they decrease when the iteration number grows;
- the number of nodes could change (limited by an upper bound) as well as the possible connections. They can be added or deleted during the learning phase. This is an advantage compared to the SOM, because now the network is more adaptable to the variability of the manifold.

  In the GNG, a local error is associated to each reference vector. In this way, when a new node must be added to the network, the error information is used to find the best position that reduces the global error. On the other hand, in the SOM the number of nodes and the connections between them are strongly constrained;
- some topology functions can be used to estimate the adaptability of the network to the manifold.

4.3.1. Algorithm Description

In literature, the GNG is known for its property to correctly represent a variable manifold. For example, during time, if the input distribution changes, the system takes this information into

account by updating the number of nodes, their positions and connections. This feature is implemented in the most important steps of the GNG described in the Pseudocode 4.1.

In (p.1) the network is initialized with the minimum number of nodes, in this way the configuration is as simple as possible. During the network training, the GNG will find automatically the best configuration. In (p.2) the input signal is selected, in particular for the application proposed in Section 5.3.2.3, each element of the manifold has the same probability to be used. In (p.3) the competitive rule is implemented, its aim is to find the two closest nodes to the input.

In (p.6) the network is modified taking into account the shape of the manifold, indeed the BMUs are attracted by the input signal. ε_w and ε_n must be in the range $[0-1]$, depending on how fast the adaptation must be. The displacement is linear and proportional to the distance between ξ and w_{s1} (or w_{s2}). However, in literature there are some works where the adaptation scheme is not linear, for example it could implement gravity-inspired displacements [153].

The algorithm is able to manage some situations where the network is not able to follow rapid changes in the manifold. To this end, the condition in (p.4) is implemented to take into account if a connection is no longer necessary. An edge is deleted if its age is greater than a specific value, as stated in (p.8). Two possible situations justify this operation:

- there is no more correlation between the nodes of an old edge, for example due to an alteration of the manifold or a possible intermediate node;
- a node is too far from the manifold and then it is deleted from the network because never selected as BMU.

The edge between s_1 and s_2 is reset by the condition in (p.7). In this way the connection between active nodes is reinforced and the probability to be deleted in the future is reduced. To each node is associated a reference vector and an error called *accumulated error*. It is increased by the equation in (p.5) and it is used to have information about how much space in the manifold is covered by each node. For example, a big error means that a specific node is chosen frequently as BMU and then a large amount of the manifold is mapped to it. The (p.10) is used to reduce all the errors at each step of the GNG. In this way, when the algorithm is implemented, possible overflow situations of the variables that store Err are avoided.

After λ inputs, in (p.9) the algorithm adds a new node in the network. The nodes with the highest error (q) and its neighbour (f) are identified. To reduce the global error of the network, the new node is placed between q and f. The connections are properly updated as stated in (p.9c). Finally, in (p.9d) the error of q and f are decreased. Therefore, next time that a new node is placed, another pair of nodes will be selected.

In literature, alternative solutions add a new node not only after λ inputs, but when it is necessary [154]. For example, this is possible monitoring the trend of a global error and, when it passes a specific threshold, the algorithm adds a node.

Another modification that concern λ has been proposed by Rodríguez et al. [155]. They modified the original algorithm enabling to add more than one node every λ iterations. The result is a model than fits the manifold faster than the standard GNG.

Algorithm 4.1 GNG algorithm.

1: Initialize the model with two nodes at random positions in the manifold;

2: Select an input element (ξ) from the manifold using a defined probability;

3: Find the nearest node (s_1) and the second-nearest node (s_2) with respect to ξ. These nodes are the BMUs:

$$s_1 : \|\xi - w_{s1}\| \leq \|\xi - w_i\| \ \forall i \in N$$
$$s_2 : \|\xi - w_{s2}\| \leq \|\xi - w_i\| \ \forall i \in N \ \cap s_1$$

4: Increase the age of all edges leaving s_1;

5: Increase the accumulated error associated to s_1:

$$Err(s_1) = Err(s_1) + \|w_{s1} - \xi\|^2$$

6: Move w_{s1} and its direct topological neighbours (nodes connected to s_1) towards ξ:

$$w_{s1} = w_{s1} + \varepsilon_w(\xi - w_{s1})$$
$$w_{sn} = w_{sn} + \varepsilon_n(\xi - w_{sn})$$

where ε_w and ε_n are the weight coefficients of the distance between ξ and w_{s1} or the neighbours of w_{s1}, respectively. Following this procedure, the displacement is directly related to the distance.

7: If s_1 and s_2 are connected by an edge, set the age of this edge to zero. Otherwise, if such an edge does not exist, create it;

8: Remove edges with age higher than a_{max}. If as a consequence it gives a node without edge, delete also this node;

9: After λ input elements, add a new node in the network as follows:

a Find the node (q) with the maximum accumulated error;

b Add a new node (r) between q and its neighbour (f) with the largest accumulated error;

$$w_r = 0.5(w_q + w_f)$$

c Connect the new unit r with nodes q and f, and remove the original edge between q and f;

d Decrease the accumulated error of q and f by multiplying them with a constant α. Initialize the error variable of r with the new value of the error variable of q.

$$E_q = E_q - \alpha E_q$$
$$E_f = E_f - \alpha E_f$$
$$E_r = E_q$$

10: Decrease all the accumulated errors by multiplying them with a constant β;

$$E_c = E_c - \beta E_c \quad (for\ each\ node\ c)$$

11: If a stopping condition is not yet fulfilled, go to step 2.

4.3.2. Topology Preservation

The *topology preservation* takes into account the network structure to measure how well the manifold is represented. It can be exploited while the network is fitting to the manifold to assess if the graph continues to be consistent with the input or not. Many techniques have been proposed in literature to solve this task, but the most important are the following:

- *Topographic product* [156]: it uses the distance between each nodes pair, both in the network (output space) than between reference vectors (input space).

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log \left(\left(\prod_{l=1}^{k} \frac{d^M \left(w_j, w_{n_l^A(j)} \right)}{d^M \left(w_j, w_{n_l^M(j)} \right)} \cdot \frac{d^A \left(j, n_l^A(j) \right)}{d^A \left(j, n_l^M(j) \right)} \right)^{1/2k} \right) \tag{4.18}$$

  Where j is a neuron, w_j is its reference vector and $n_l^M(j)$ is the l^{th} neuron close to j found using the Euclidean distance d^M defined in the input space. Whereas $n_l^A(j)$ is the l^{th} neighbours close to j measured in the output space with d^A.

  Instead of the Euclidean distance d^M, Flórez-Revuelta et al. in [157] introduced the geodesic distance. It is defined as the shortest path inside the manifold that connects two reference vectors. Thanks to this improvement, the limitations of the *topographic product* are overcome because now the characteristics of the manifold are considered;

- *Topographic function* [158]: it uses the function Φ_w to map a point in the input space to the output space. A network preserves the topology if close nodes in the output space have close reference vectors in the input space and vice versa. The formula of the *Topographic function* also considers the information of the manifold.

A network that preserves the topology is called *Topology Preserving Network* (TPN). The GNG has this property because it is able to change the number of neurons and the connections between them in a very flexible way. Whereas, for SOM and SOM_Ex, it is necessary that the network is chosen with an adequate criterion, because it cannot be modified while the algorithm is running.

Chapter 5.

Applications

The information provided by the camera sensors and the wearable device described in Chapter 2 is now exploited as input data to three AAL applications.

The first one is a fall detection algorithm that uses the depth stream provided by Kinect V1 installed on the ceiling. When a person is inside the monitored area, the system recognizes him among all the blobs and tracks its centroid's height to detect a fall. A similar application has been devised using the skeleton stream of Kinect V2 in frontal view, along with the acceleration measured by two SHIMMER devices set to the belt and the wrist of the user. In this case, the synchronization results introduced in Section 3.1 are exploited with the aim to use at the same time the raw data measured by the sensors. The result is a data fusion application that improves the performances of the separate solutions.

The second application uses a wearable device worn over the clothes near the chest by an elastic belt and Kinect V2 placed in front of the person. The aim is to measure kinematic parameters of the body while the subject performs the assessment tool called TUG test.

Finally, the last application will monitor the drink intake movements of a person during a meal, using depth and colour streams framed by Kinect V1 placed on the ceiling. The three self-organizing networks, introduced in Chapter 4, are tested with in input the person's PC calculated from the depth frame. The SOM-SOM_Ex are used to fit a pre-defined model to the PC, whereas the GNG directly provides in output an adapted network. The three techniques are compared to find the solution that tracks the person's movements with the best accuracy. Afterwards, the spatial synchronization (Section 3.2) between depth and RGB frames allows focusing the research to useful objects like dishes and glasses placed on a table where the person is eating and drinking.

5.1. Fall Detection Algorithms

In this section, two approaches to tackle the problem of the fall detection inside a home environment are described. The first solution directly uses the depth frame provided by Kinect V1 installed on the ceiling to monitor different people inside the covered area and to trigger an alarm when a risk situation is recognized. In addition, more than two devices could be used in an array configuration to expand the monitored area. Thus, a central Pc will receive a depth stream for each sensor connected to the system. In order to reduce the load of the network, a depth compression solution is described at the end of the first algorithm.

In the second approach, the skeleton provided by Kinect V2 and the acceleration measured by the wearable devices are combined together in a data-fusion application. The aim is to classify the activities performed by the user in two main groups: fall and ADL. In particular, three algorithms are tested considering different sensor configurations and data analysis.

5.1.1. Depth Frame as Input Data

This application exploits the depth stream, at a resolution of 320×240 pixel, provided by Kinect V1, placed on the ceiling at a height of 3 m (*MaxHeight*) from the floor. In this configuration, the monitored area is 8.25 m^2. Compared to other sensor set-ups proposed in literature, such as the wall configuration in [39, 40], here the monitored area is reduced, but an evident advantage is the overcoming of possible partial occlusions caused by furniture between the subject and the camera.

Another noticeable feature of the depth stream is that the privacy of the monitored person is respected, indeed in the depth map his identity cannot be inferred from the distance characteristics of his shape. This result can be reached also with an RGB camera, but a pre-processing phase is necessary to hide the human blob in the colour frame. Moreover, the variability of the environment illumination does not affect the depth stream as much as the RGB information. Therefore, the application can be used also during the night when light conditions are limited.

The devised algorithm can be divided in the following steps:

- *pre-processing phase and segmentation*: the algorithm fills the zero-pixels of each captured raw frame and uses an edge detection solution to highlight the object's contours. A reference frame (RF) is compared with the processed frame to highlight the blobs in foreground;
- *blob labelling*: each foreground pixel is labelled with the index of the belonging blob;
- *person identification*: an ad-hoc algorithm finds if blobs in the scene correspond to a person or to an object;
- *person tracking*: each person is tracked and possible fusions between blobs are taken into consideration by the system;
- *fall detection*: the algorithm is able to identify when a person is lying on the floor, analysing the height of the blob per each frame. In this section, two different fall situations are tested to show the performance of the algorithm.

The dataset *Dts_Fall*, available in [122], has been used to test the application. Four volunteers, aged between 26-27 years and height in 1.62-1.78 m, have been recruited for a total of 20 tests. The dataset is divided into two groups: in the first one (first 10 sequences), tracking performances of the algorithm are tested with different people walking under the sensor whereas in the last 10 sequences, two people fall while they are monitored by the sensor.

Figure 5.1 shows the pipeline of the algorithm. The distance values of depth frames are encoded in colour values, warm colours are used for further objects and when the surfaces are close to the

sensor the colour changes to cold values. The dark blue is associated to the pixels with distance value equal to zero, i.e. points whose depth has not been provided by the sensor.

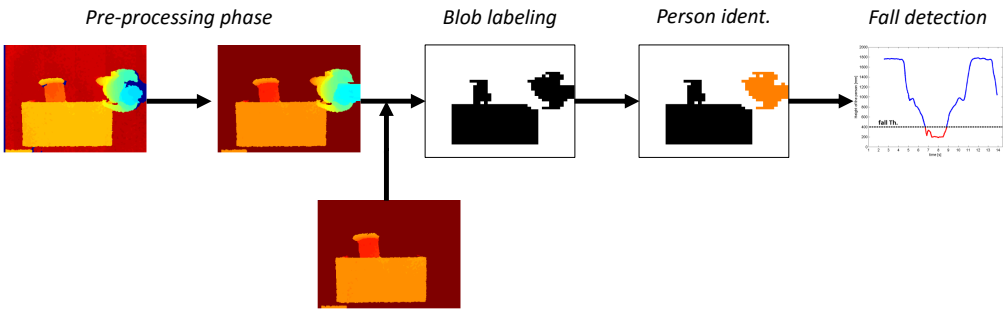


Figure 5.1.: Pipeline of the fall detection algorithm. Original depth frame, floor pixels equalization and zero pixel filling, blob labelling and person recognition, monitoring of the person distance from the floor.

5.1.1.1. Pre-processing Phase and Segmentation

The raw frame provided by the sensor must be pre-processed before recognizing possible objects in the monitored area. In particular, the pixels close to the floor in the interval of 20 cm from the surface are set to *MaxHeight*. Another operation to perform on the depth frame is the substitution of zero pixels. As introduced in Section 2.1.3, when Kinect is not able to calculate the depth distance, it assigns the value 0 to these pixels. The algorithm will replace the point with a valid depth value. In particular, within the same row, the system scans both the left and the right adjacent elements to find the first non-zero pixel.

At this point, the processed frame (CF) is compared to the RF and depth pixels belonging to the foreground are emphasized. The new frame is called foreground frame (FF). As visible in Figure 5.1, the RF is very similar to the CF, except for the pixels that belong to a person. It is recorded by the system only when nobody is under the sensor.

If depth values in CF differ from the corresponding points in RF, they are incremented of the quantity $Depth_{Gap}$, as defined in Eq. (5.1). To understand if a pixel is in foreground, its depth value is compared with the same pixel in RF. When the difference is higher than Th_{Per} (50 mm), it means that a new person enters in the scene. This processing on CF has the same meaning of the Intensity Level Slicing process [159] for an RGB frame. It is used in the RGB frame to enhance colour differences, while in our case Eq. (5.1) will improve the separation between different objects.

$$FF(u,v) = \begin{cases} CF(u,v) + Depth_{Gap} & if |CF(u,v) - RF(u,v)| > Th_{Per} \\ CF(u,v) & otherwise \end{cases} \tag{5.1}$$

The Sobel edge detection algorithm (Section A.2) is used to find the blob's borders in FF and then the depth values of these points are set to *MaxHeight* (Eq. (5.2)). The operation improves the

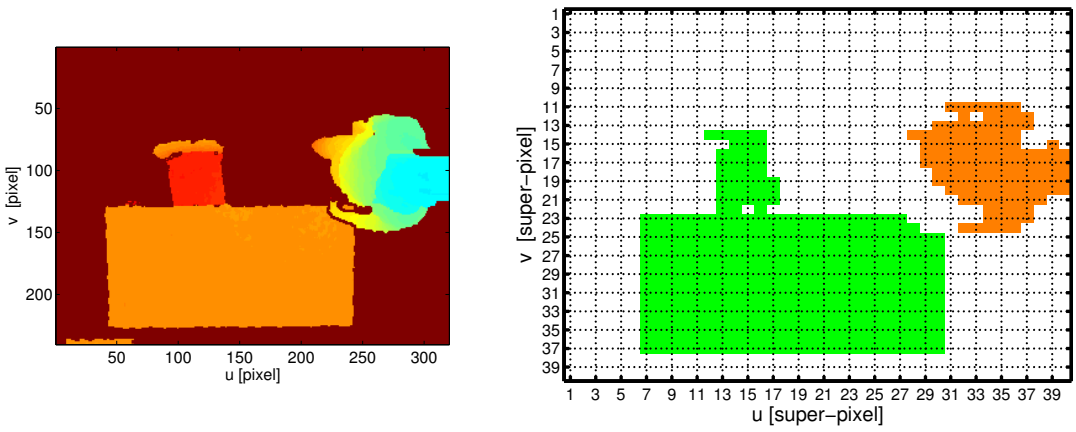
separation between the elements in foreground, for instance when they are overlapped. Therefore, there will be an improvement in the performance of the labelling process.

$$FFSobel(u,v) = \begin{cases} MaxHeight & if\ Sobel(CF(u,v)) > Th_{Sobel} \\ FF(u,v) & otherwise \end{cases} \tag{5.2}$$

After the edge detection algorithm, in Figure 5.2a the result is visible. In particular, the left arm of the person is separated from the table by the red contour, while in FF the two blobs were connected. Finally, before using the labelling algorithm, the frame is down-sampled of 8×6 factor and the new matrix is called super-pixel foreground frame (FFs). Each super-pixel is set to 1 if all the depth values inside it are different from *MaxHeight*, otherwise the super-pixel is set to 0.

This operation has two advantages, it improves the separation between the blobs and drastically reduced the computational time because the labelling algorithm will get in input FFs (40×40 elements) instead of $FFSobel$ (320×240 elements).

The result of the down-sampling operation is shown in Figure 5.2b. Compared with $FFSobel$, the blob of the person is further separated from the table.



(a) (b)

Figure 5.2.: The output of Sobel algorithm is used to separate blobs in the depth frame (a). $FFSobel$ after the down-sampling (b).

5.1.1.2. Blob Labelling

The binary image FFs is given in input to a single pass connected component labelling algorithm. It finds all the connected blobs and provides in output the matrix Mat_Obj, where each row stores the super-pixels coordinates that belong to a single blob. For example, the output of the labelling algorithm for the FFs in Figure 5.2b is the Mat_Obj with the first two rows filled with the coordinates of the two coloured blobs in the image.

The algorithm scans one row of FFs at a time, from the top to the bottom, to find blob portion (BP) of the same object. When there is a null super-pixel between two BP, they are classified as separate. The details related to each BP, stored in the variable Mat_RowObj, are the following:

- first column index of BP;
- last column index of BP;
- identifier of the object (OI) within the Mat_Obj structure.

When the algorithm evaluates the next row of FFs, it compares the new Mat_RowObj variable with previous ones. If there is just a partial overlapping between the BPs inside two Mat_RowObj, it means that the BP found in the current row belongs to the same blob. For example, looking at Figure 5.2b, Mat_RowObj for rows 11 and 12 is equal to:

$$Mat\_RowObj(11) = \begin{bmatrix} 31 & 36 & 1 \end{bmatrix} \qquad Mat\_RowObj(12) = \begin{bmatrix} 32 & 32 & 0 \\ 34 & 37 & 0 \end{bmatrix}$$

The third elements in $Mat\_RowObj(11)$ means that the BP belongs to the object 1, whereas the values 0 in $Mat\_RowObj(12)$ imply that the BPs have not been classified yet. Since the two elements in row 12 have respectively columns 32 and 34-35-36 in common with the BP of the previous row, they are assigned to the object 1.

Table 5.1 shows the coordinate values stored in Mat_Obj after the processing of row 14. Due to width limitation of the page, only the first five coordinates per object are shown.

OI	**u**	**v**	**u**	**v**	**u**	**v**	**u**	**v**	**u**	**v**
1	31	11	32	11	33	11	34	11	35	11
2	12	14	13	14	14	14	15	14	16	14

Table 5.1.: Object coordinates for each blob found by the clustering algorithm.

The algorithm is also able to consider when two separate BPs merge in the same blob (side-effect). Again, in row 15 of Figure 5.2b, there are:

$$Mat\_RowObj(14) = \begin{bmatrix} 12 & 16 & 2 \\ 28 & 36 & 1 \end{bmatrix} \qquad Mat\_RowObj(15) = \begin{bmatrix} 14 & 16 & 0 \\ 29 & 36 & 0 \\ 39 & 39 & 0 \end{bmatrix}$$

As before, the first two BPs in $Mat\_RowObj(15)$ are assigned respectively to the object 2 and 1 because there is a superposition with the previous BPs in $Mat\_RowObj(14)$. Alternatively, the third element in $Mat\_RowObj(15)$ does not match with any one of the identified objects. Therefore, the algorithm classifies this point as a new blob (OI=3). Only in the next row, the

system is able to correct this situation.

$$Mat\_RowObj(15) = \begin{bmatrix} 14 & 16 & 2 \\ 29 & 36 & 1 \\ 39 & 39 & 3 \end{bmatrix} \qquad Mat\_RowObj(16) = \begin{bmatrix} 13 & 16 & 0 \\ 29 & 40 & 0 \end{bmatrix}$$

It finds out that the second BP in $Mat\_RowObj(16)$ is connected to two different objects (OI=1, OI=3), so that the algorithm joins together these two elements in a single blob. In the same way, the identical principle is used when there are more than two BPs to merge.

The fundamental steps of the labelling algorithm are summarized in Pseudocode 5.1.

Algorithm 5.1 Blob labelling.

Input: $FFs\ frame$
Output: $Mat\_Obj\ matrix$

```
1:  for each row of FFs do
2:    Filling of Mat_RowObj structure
3:    for each element in Mat_RowObj do
4:      if FFs row index > 1 then
5:        if Mat_RowObj(row) and Mat_RowObj(row − 1) are overlapped then
6:          connect the actual BP to the corresponding previous object
7:        else
8:          classify the actual BP as a new object
9:        end if
10:     else
11:       classify the actual BP as a new object
12:     end if
13:     if side-effect then
14:       Mat_Obj reordering
15:     end if
16:   end for
17: end for
```

5.1.1.3. Person Identification

After the labelling procedure, for every i^{th} blob found, the algorithm calculates the central super pixel (scp_i) as the closest blob's super-pixel to the centroid. It constrains the central point to be inside the blob and, consequently, the tracking solution presented in the next section will be improved. The equivalent pixel in the original frame is called cp_i.

Another point calculated for each blob is its highest point, or alternatively, the pixel with the closest distance to the sensor (mp_i).

The algorithm recognizes if a blob is a person or not, exploiting the anthropometric characteristics of the human body [160]. The features controlled are:

- distance between the head and floor;
- gap between the head and the shoulder;
- head ratio.

Usually, when a blob is a person, mp_i is placed in the centre of the head and, for this reason, it is exploited as starting point to calculate the three features. Analysing the mp_i coordinates in the CF, the algorithm evaluates the depth information moving towards the eight cardinal directions (N, NE, E, SE, S, SW, W, NW). As visible in Figure 5.3a, mp_i is highlighted with a red dot and the search paths evaluated by the algorithm are represented with black lines. The system calculates the difference between two consecutive depth values ($DiffCons$) at every pixel step. Starting from mp_i, the black lines stops in the first pixel that has the depth value equal to $MaxHeight$ or when $DiffCons$ verifies one of the next two conditions.

Head-Ground Distance Gap

Due to the sensor configuration, it is possible that a considerable depth gap between two consecutive points can be verified if moving from the centre of the head to an external direction. This feature is typical of the human shape and it is implemented by the following condition:

$$DiffCons > CF(u_i, v_i)/2 \qquad with\ (u_i, v_i) = mp_i \tag{5.3}$$

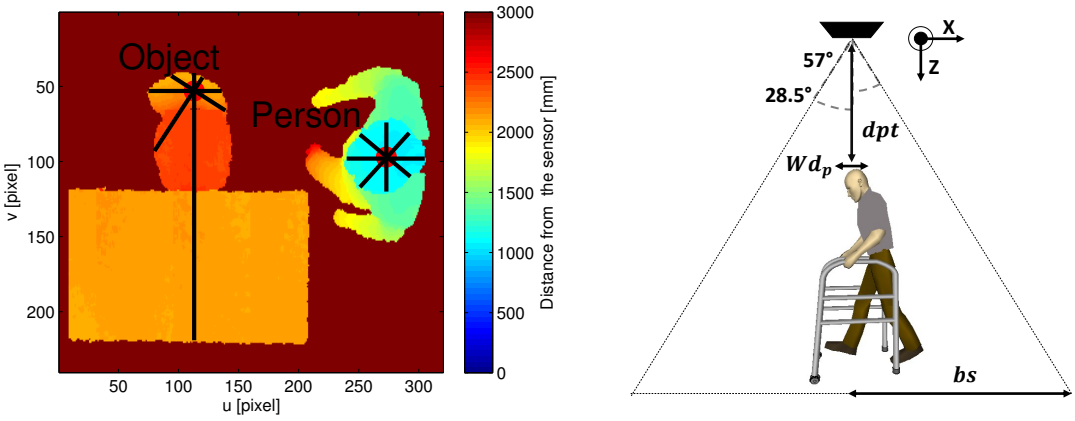
Head-Shoulder Distance Gap

The gap between the head and the shoulder is another feature that characterizes the human shape. The tests performed in [160] provide a height of the head in the range 20-30 cm. This value is calculated as the difference between the number 805 and 841 in [160], i.e. the human and the shoulder heights, respectively. At least, two directions out of eight must verify the gap.

Head Ratio

In the third feature, the diameter of the head must be in the range 20-40 cm which has been calculated thanks to specific test campaigns indicated by numbers 441 and 702 in [160].

To calculate the real diameter of the head, the algorithm uses the segments starting from mp_i and ending in the pixel coordinates that verify the previous two features. Each segment is added together with its opposite one to obtain the four principal directions (N, NE, E, SE). These pixel values must be converted in real dimensions exploiting the depth sensor characteristics and the



(a) (b)

Figure 5.3.: Person identification in the CF (a) and side view of system set-up (b).

proportion in Eq. (5.4).

$$\frac{wd_p}{320} = \frac{wd_r}{2bs} \rightarrow wd_r = \frac{2bs \cdot wd_p}{320}, \qquad bs = dpt \cdot \tan(28.5°) \tag{5.4}$$

Where, as visible in Figure 5.3b, wd_p is the pixel length of the segment in the horizontal direction. dpt is the distance between the sensor and the mp_i, 320 is the total number of pixels per row.

Finally, 28.5° is half FOV of the sensor in the horizontal plane and wd_r is the real length of the segment. If the segment is parallel to the v direction, 320 must be replaced with 240 and 28.5° with 21.5°.

When the segment is parallel to NE or SE directions, the real distance is equal to the combination of its orthogonal components:

$$wd_r = \sqrt{\left(\frac{2dpt \cdot \tan(21.5°)wd_p}{240}\right)^2 + \left(\frac{2dpt \cdot \tan(28.5°)wd_p}{320}\right)^2} \tag{5.5}$$

When a blob is marked as person, the system stores its most important attributes in the $CoordPerson$ data structure:

- cp_{Per}: pixel coordinates of cp_i;
- scp_{Per}: super-pixel coordinates of cp_i;
- mp_{Per}: pixel coordinates of mp_i;
- smp_{Per}: super-pixel coordinates of mp_i.

Otherwise, the blobs is classified as object and its details are saved in $CoordObj$:

- cp_{Obj}: pixel coordinates of cp_i;

– scp_{Obj}: super-pixel coordinates of cp_i;

– mp_{Obj}: pixel coordinates of mp_i;

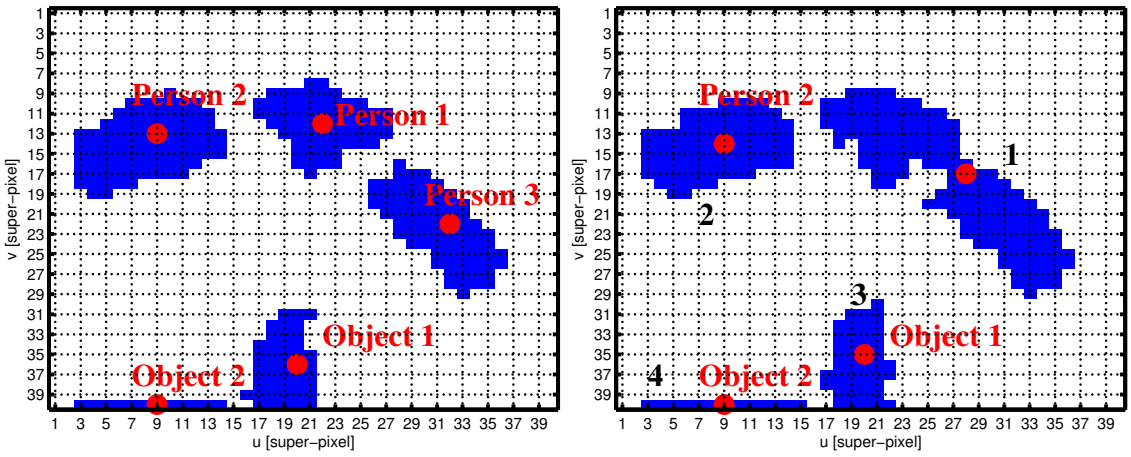
– smp_{Obj}: super-pixel coordinates of mp_i.

5.1.1.4. Person Tracking

The person identification function, described in the previous section, is used only for those blobs which are not classified as person yet. The person's blob has to be tracked in the scene when the subject is performing different movements. For this reason, when the sensor provides the k^{th} frame, the algorithm must correlate the found blobs with the others already classified in the $(k-1)^{th}$ frame.

One solution for each blob labelled in the $(k-1)^{th}$ FFs, is to search its closest cluster in k^{th} FFs, comparing the points scp calculated for $(k-1)^{th}$ FFs with the others in k^{th} FFs. Unfortunately, this approach does not work if the frame rate decreases or when two or more blobs are fused in a single element. Indeed, the edge detection algorithm, used in Section 5.1.1.1, separates very well the objects only if there is a relevant depth difference between the blob's borders. Unfortunately, if two people are very close, the depth gap is less noticeable and they are fused in a single cluster. The proposed tracking solution takes into account this aspect and force the separation when it recognizes a fusion.

Initially, the algorithm places the scp points calculated in the $(k-1)^{th}$ FFs, into the k^{th} FFs and assigns each blob of the new frame to the specific person or object previously classified. For example, as visible in Figure 5.4a at $(k-1)^{th}$ FFs, the system has identified three people in the



(a) (b)

Figure 5.4.: $(k-1)^{th}$ FFs (a) and k^{th} FFs (b) where a fusion situation affects the blobs of two people.

centre of scene and two objects close to the lower border. They have the following super-pixel coordinates:

$$People: \; scp_{Per}(1) = (22, 12) \qquad scp_{Per}(2) = (9, 13) \qquad scp_{Per}(3) = (32, 22)$$
$$Object: \; scp_{Obj}(1) = (20, 36) \qquad scp_{Obj}(2) = (9, 40)$$

In the next frame the corresponding FFs is shown in Figure 5.4b. Now, there are four blobs identified by the labelling algorithm. The *scp* points of person no.2 and objects no.1-2, classified by the system in $(k-1)^{th}$ FFs, are placed in the k^{th} FFs and the blob 2 is marked as person while blobs 3 and 4 as objects. The person identification algorithm checks the object no.1-2 and the attributes of each person/blob are finally updated.

The system solves fusion situations evaluating the $TrackingInfo$ data structure (Table 5.2). It is a matrix with as many rows as blobs found in the k^{th} FFs. $TrackingInfo$ stores the information for the r^{th} row as follows:

- $Pers$: identifiers (given at $(k-1)^{th}$ FFs frame) of the person found in the r^{th} blob;
- Obj: identifiers (given at $(k-1)^{th}$ FFs frame) of the objects found in the r^{th} blob;
- N_Pers: number of elements in $Pers$ field;
- N_Obj: number of elements in Obj field.

Blob index	**Pers**	**Obj**	**N_pers**	**N_obj**
1	1-3	-	2	0
2	2	-	1	0
3	-	1	0	1
4	-	2	0	1

Table 5.2.: $TrackingInfo$ data structure for k^{th} FFs filled with the person/blob identifiers of $(k-1)^{th}$ FFs.

The combination of values stored in the fields N_Pers and N_Obj leads to one of the following conditions:

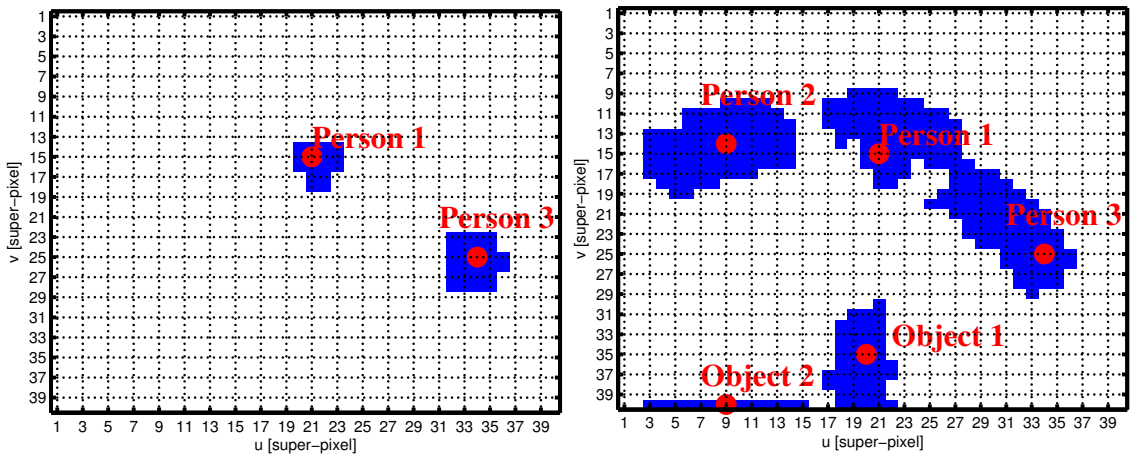
A $N\_Pers == 0$ && $N\_Obj \leq 1$
No person is found in the specific blob and, if N_Obj is equal to 0, the blob represents a new object. On the contrary, If N_Obj is equal to 1, an "old" object classified in $(k-1)^{th}$ FFs is now found in the specific blob. In both the situations, the person identification algorithm is used to assess if the blob is a person.

B $N\_Pers == 1$ && $N\_Obj == 0$
In this case, an "old" person is found in the current blob and there is no fusion.

C *Otherwise*
Differently from the two previous situations, now a fusion occurred and in the current blob there are two or more objects/people. The algorithm will separate each element with the goal to continue the tracking.

In connection with the previous description of Figure 5.4b, blob 2 complies the condition B, while blobs 3-4 verify situation A. Therefore, in both the cases, the fusion is not present. Whereas, for blob 1 the algorithm is able to find a fusion, because there are two people in the same cluster, as confirmed by the first row of $TrackingInfo$.

In this last case, to update the $CoordPerson$ for person no.1-3, the system will force the separation between the connected blobs. In particular, it discards the super-pixels not close to the *smp* coordinates, as visible in Figure 5.5a, and uses the new foreground super-pixels to calculate the positions of the person no.1-3. This operation is also implemented when more than two people/objects are fused together.



(a) (b)

Figure 5.5.: Blob separation after fusion situation (a), final result for k^{th} FFs (b).

The result is shown in Figure 5.5b, where all people and objects found in the $(k-1)^{th}$ FFs have been linked to new blobs in the k^{th} FFs.

The tracking performances of the system have been successfully evaluated with the first group of tests in the dataset Dts_Fall. In particular, some people entered the scene at different times and walked in the monitored area. The application has been able to recognize and track each person, taking into consideration possible fusion situations for the entire length of the tests.

5.1.1.5. Fall Detection

When the system recognizes a person, it monitors the movements inside the scene to understand if a fall occurs. In particular, the height information in cp_{Per} is evaluated in every frame and the

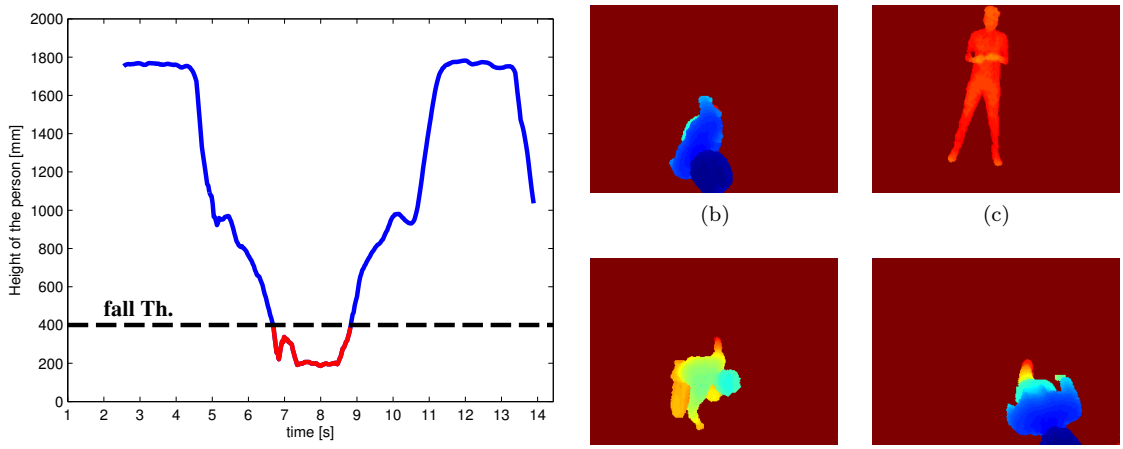
algorithm triggers an alarm when the distance is close to 400 mm (Th_{fall}). This value has been experimentally found and it is similar to the thickness of the body when the person is lying on the floor. The second group of tests in Dts_Fall, has been used to assess the performance of the fall detection application. In particular, the subject enters in the scene and simulates a fall (simple falls). On the other hand, he interacts with objects such as tables or chairs, and at the end, he falls (complex fall).

Figure 5.6a shows the trends of cp_{Per} in the simple case. The person does not interact with objects, but just enters in the monitored area and falls. In particular, at the beginning of the test, nobody enters in the monitored area, whereas after two seconds a person is recognized by the system.

He stops in the centre of the frame and falls between 5 and 9 s. The system triggers an alarm when the height of cp_{Per} decreases under Th_{fall}. Finally, he gets up and leaves the area. When he is close to the border of the frame, the height of cp_{Per} decreases again because only lower parts of the body are visible to the sensor. From Figure 5.6b to Figure 5.6e are shown the four most important CFs for the first example of fall.

The second example is depicted in Figure 5.7. Now the person interacts with some objects before the fall. In particular, he enters in the monitored area and sits close to the table. When he stands up, a fall occurs. Finally, after the fall, the person gets up and leaves the scene.

Differently from the previous situation, now the reference frame is fundamental to keep the blob of the person separated from the table. Indeed, the modification of person's depth values with Eq. (5.1) allows to the Sobel algorithm detecting the volunteer's borders also when he is sitting. In



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

Figure 5.6.: Height trend of cp_{Per} during a simple fall situation where the person falls without interaction with objects (a). Sequence of the most important CFs for a simple fall: the volunteer enters in the scene (b), falls (c), gets up (d) and leaves the monitored area (e).

this way, the labelling algorithm can constantly find both the person's cluster and the table.

The complete system has been implemented in C++ language and executed in a desktop Pc featuring Intel i5 processor, and 4 GB RAM with Windows 7 as operating system. The depth resolution is set to 320×240 pixels with a frame rate of 30 fps.

The system has also tested on an embedded platform (Odroid-X2 development version) featuring Ubuntu Linaro 12.11 operating system, 2 GB RAM and ARM Cortex-A9 1.7 GHz Quad Core architecture processor. In this case, due to the limited computational characteristics of the board, the proposed solution works at a frame rate lower than 30 fps. However, the algorithm still works in the correct way.

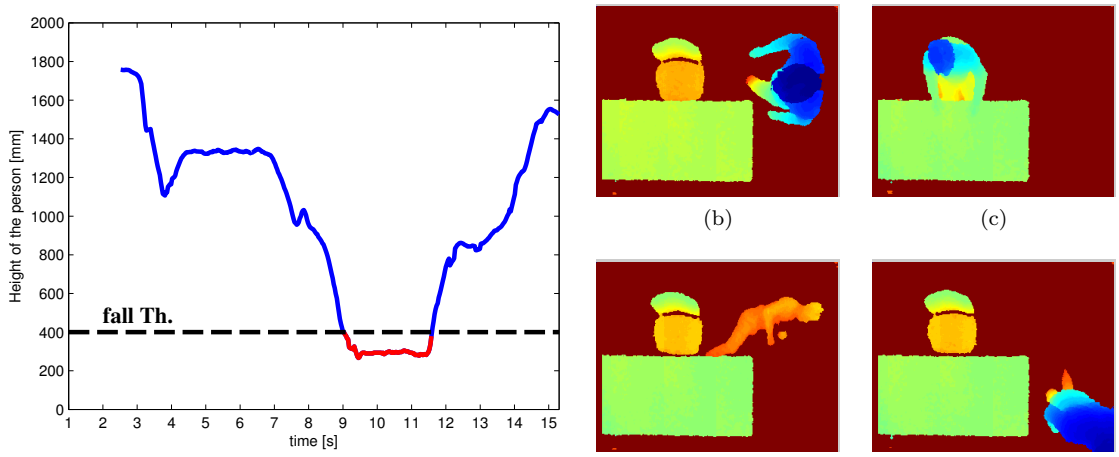


Figure 5.7.: Height trend of cp_{Per} for a complex fall (a). Sequence of the most important CFs for the second example of fall: the volunteer enters in the scene (b), sits near the table (c), falls (d), gets up and leaves the monitored area (e).

5.1.2. Depth Streaming and Compression

When the monitored area by the fall detection algorithm must be extended, a solution is to use two or more Kinect sensors at the same time and devise a sophisticated algorithm that handles the person tracking in the overlapped areas.

Another issue that has to be solved in a multi-camera application is the depth stream handling from all the sensors to the central system, which runs the fall detection algorithm. Two solutions are possible: the cameras are directly connected through USB cables to a Pc that uses the received streams or sends them over the external network to another control system.

This section illustrates the problem of streaming the depth information over the network. It describes the application that handles the communication between the devices and the compression solution used to reduce the bandwidth demand by a raw depth stream.

The devices used are a desktop Pc (server) connected to the network and the embedded board Odroid X2 (client), introduced in the previous section, which controls Kinect V1. In this solution, it is not possible to use the official Microsoft SDK, because the only operating systems supported by the platform are Ubuntu Linaro 12.11 and Android. Therefore, the official Kinect SDK cannot be installed and the library Libfreenect [108] has been selected as possible alternative drivers to control the Kinect sensor.

The idea is to stream both the RGB and the depth data using socket connections [161] based on the TCP protocol. It has been chosen, instead of the UDP, for its connection-oriented properties. The advantages are:

- automatic retransmission of the lost packets through acknowledgement (ACK) solution;
- packet organization solution;
- error correction using a checksum fields.

As visible in Figure 5.8, the implemented solution uses three different sockets:

- socket 1: open and close streaming communication;
- socket 2: stream of the depth frame;
- socket 3: stream of the RGB frame.

In details, the devices are connected to the network with an assigned IP address and the server knows the address of the client. Each socket uses a dedicated port (Sck1:36000 Sck2:18000 Sck3:10000). The server sends a command to the client to start the communication, then it waits on reserved ports the possible data from the client. The Odroid board, which is listening on the port Sck1, receives the command and starts recording data from the camera. It splits the depth/RGB frames in more packets that will be transferred over the network. These data will be available on the server ports Sck2 and Sck3, respectively. When the communication with the client must be closed, the server sends a command to switch-off the data streaming.

The streams sent over the network need a considerable bandwidth if they are sent in a no compressed format. For example, with a resolution of 320×240 pixel, considering 2 bytes per pixel and a frame rate of 30 fps, the depth frame requires approximately 4.6 MB/s without the TCP/IP overhead. Therefore, it is necessary to reduce the amount of information using a compression algorithm.

For the RGB stream, a solution is the JPEG compression format, whereas for the depth stream there are no standardized compression formats. Most of the compression algorithms used for colour need 8 bit three channels stream as input. As described in Section 2.2, the depth format is a 16 bit single channel stream, then a modified solution for the compression is required.

In literature, some applications propose encoding 16 bit depth format in 8 bit RGB [162]. In this way, it is possible to use the standardized compression algorithms used for the colour stream, but the resolution of the depth stream will be reduced.

The proposed solution uses a modified version of the JPEG algorithm and the steps are the following:

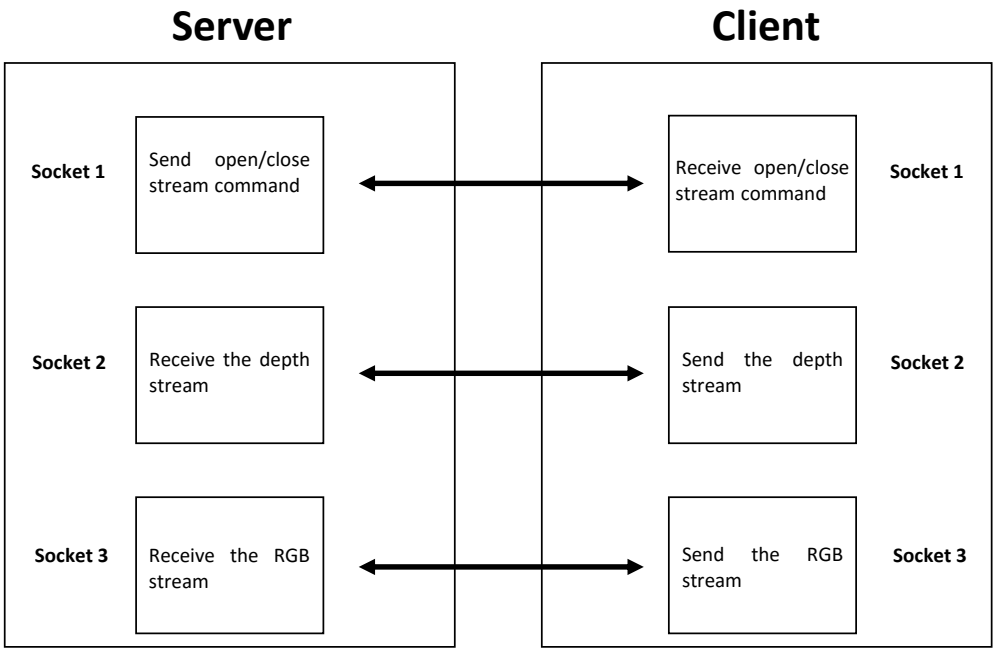


Figure 5.8.: Description of the socket based architecture. The socket 1 is used for streaming management whereas the other two are used to send and receive both depth and RGB streams.

- the frame is divided in blocks of 8×8 pixels;
- for each block is calculated the Discrete Cosine Transform (DCT);
- quantization of the DCT coefficients;
- Entropy encoding on the transformed block.

The operation order is inverted for the receiver. The DCT transform is used to reduce the number of the useful elements to be transmitted. Indeed, after the conversion, the useful coefficients are in the top-left corner of the block. The other elements are close to 0 and can be discarded.

To correctly select the coefficients that must be sent, a specific mask of 8×8 pixels is multiplied by the coefficient matrix. For the JPEG algorithm, it is called *quantization matrix* and its values have been selected to preserve the most important DCT coefficients [163]. For the depth compression algorithm, the *quantization matrix* (Q_{depth}) is very similar to the previous one and takes into account the different number of bits used to represent a depth value.

To study the effects of the different quantization levels, a specific value is multiplied by the

quantization matrix. It is called *quality factor* (q_f) and depends on the *quality* parameter:

$$q_f = \begin{cases} \dfrac{200 - 2 \cdot quality}{100} & if\ quality \leq 50 \\ \dfrac{5000}{quality \cdot 100} & if\ quality > 50 \end{cases} \tag{5.6}$$

where the *quality* parameter goes from 1 to 100. High values of *quality* mean that the compressed image is similar to the original one. The final matrix is:

$$Q = q_f Q_{depth} \tag{5.7}$$

The transformed block is multiplied by Q and the values close to 0 are discarded.

In the Entropy encoding, after the "zig-zag" scanning, the Huffman coding is used to map the values in specific symbols to be transmitted. The Huffman mapping table has been built in consideration of the occurrence probability of the depth values in the dataset Dts_Fall.

As mentioned above, the operations for the receiver are inverted:

- decoding the received symbol;
- use the Inverse Discrete Cosine Transform (IDCT);
- reassemble the depth frame.

The performances of the compression algorithm are evaluated using different parameters. The first one is the *compression ratio*:

$$R_{cmp} = \frac{N.bit\ original\ frame}{N.bit\ to\ send} \tag{5.8}$$

where in $N.$ *bit to send* are counted also the bits introduced by the Entropy encoding to correctly decode the transmitted information.

The *peak signal-to-noise ratio* ($PSNR$) is another parameter used to evaluate the quality of the received image than the original one. High values indicate a better quality of the compressed frame.

$$PSNR = 20 \log \left(\frac{max(D_{tx})}{\sqrt{MSE}} \right) \tag{5.9}$$

D_{tx} is the original depth frame, and max is the maximum value in the depth frame. MSE is the *mean square error*:

$$MSE = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} |D_{tx}(u,v) - D_{rx}(u,v)|^2 \tag{5.10}$$

D_{rx} is the received frame, that is the frame at the end of the encoding, it has M rows and N columns.

Two sets of 200 frames each, called *Simple_set* and *Complex_set*, are selected from the dataset *Dts_Fall* to calculate the previous quality parameters. The frames in the first group are obtained from a simple scene, where only a desk is visible. The second group includes three people moving in the scene and some furniture.

The $PSNR$ and R_{cmp} are calculated for every single frame and, in consideration of different *quality* parameters, the mean values are visible in Table 5.3. As expected, the $PSNR$ value is high for *quality* parameter sets to values close to 100 and it decreases in the opposite case. In the first group, for *quality* equals to 20, R_{cmp} is near to 19. It means that inside the same bandwidth used to transmit the original stream, now 19 simultaneous streams can be sent.

quality	*Simple_set*		*Complex_set*	
	PSNR [dB]	R_{cmp}	PSNR [dB]	R_{cmp}
20	51,97	19,38	48,64	12,91
40	56,51	13,69	53,36	9,460
60	59,05	10,91	56,44	7,780
80	62,15	7,610	60,70	5,760

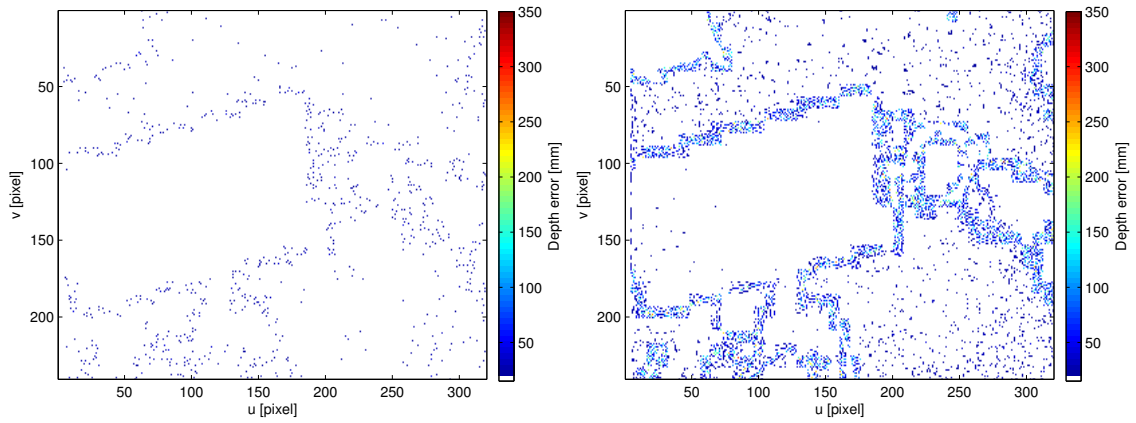
Table 5.3.: R_{cmp} and PSNR for the compression algorithm.

The *compression ratio* follows the opposite trend of the $PSNR$, the *quality* decreases for high values of R_{cmp}. In the first group, R_{cmp} for each specific *quality* value is higher than the corresponding value in the second group. This difference is due to the more depth uniformity in the first group than the second one. Indeed, for the *Simple_set*, most of the pixels have the same depth value of the floor and discontinuities are only between the table and the floor.

Figure 5.9 shows the differences between original/decoded frames for two different *quality* values. The image belongs to the *Complex_set* and inside the frame there are a table and some people. In white areas the error is equal to 0, cold colours are used for low level of errors while warm colours for significant differences.

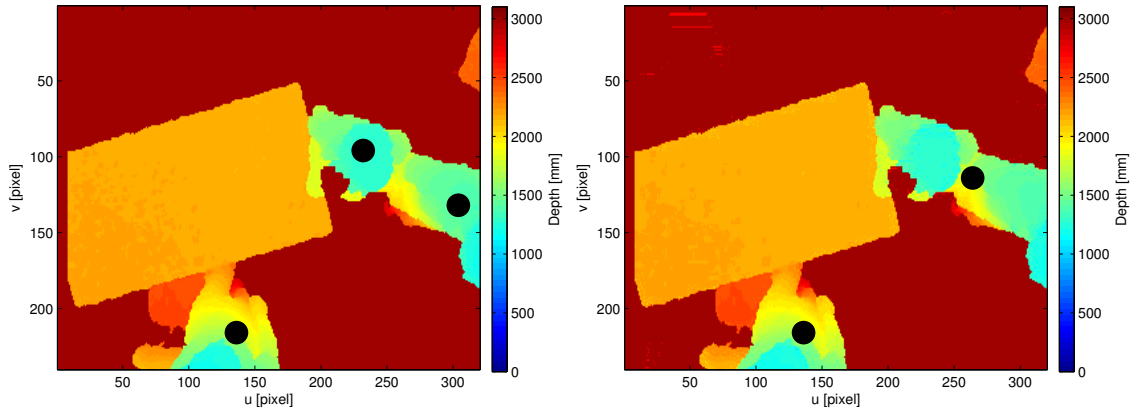
The pixels that have the most relevant errors are those close to the edges, where the discontinuities are higher than other regions. It is due to the quantization operation that removes the coefficients with high frequencies, i.e. the elements used to represent the discontinuities.

The decoded stream is used as input to the fall detection algorithm which has been described in the previous section. Different tests have implemented changing the *quality* parameter. For a *quality* of 80, the algorithm is able to identify people and track them (Figure 5.10a). The performance decreases when the *quality* is set to 20. The system continues to recognize people, but fails the tracking when two or more people are fused in the same blob. Figure 5.10b shows the situation when two people are labelled as single subject.



(a) (b)

Figure 5.9.: Error frames for the same scene, with *quality* sets to 80 (a) and 20 (b).



(a) (b)

Figure 5.10.: Results of the fall detection algorithm when decoded frames are used as input. *quality* sets to 80 (a) and 20 (b).

5.1.3. Skeleton and Acceleration as Input Data

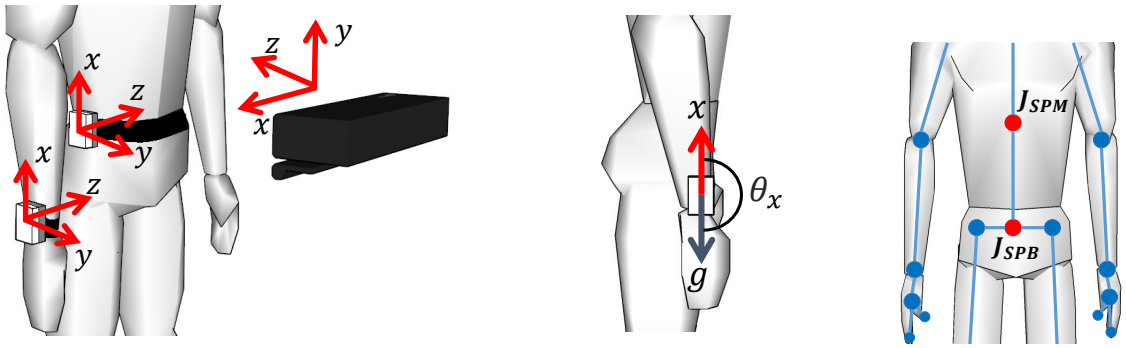
This section describes an alternative approach to the fall detection. Now the camera sensor used is Kinect V2 placed in front of the person and, instead of the raw depth frame, this time the algorithm will use the skeleton provided by the official SDK. In addition, using the synchronization solution introduced in Section 3.1, accelerations measured by two SHIMMER devices are used to devise a data fusion application.

Three different algorithms have been implemented and, after the description of their characteristics, the performances have been compared each other. The dataset *Dts_ADLFall*, introduced

in Section 2.5, will be used during the tests to classify the activities in two main groups: ADL and fall.

Figure 5.11a shows the system set-up with Kinect placed at 2 m from the person in the standard height from the floor (1.5 m). Two SHIMMER devices are placed respectively in the wrist, to emulate a smartwatch configuration, and in the belt, as suggested by Kepski et al. in [164] for a similar fall detection application.

A useful feature, obtained from the raw data measured by each axis of the accelerometer, is the magnitude M_{acc}. It is calculated using Eq. (2.16-2.17). In Figure 5.11b is visible the orientation of the x-axis of the wearable sensor with respect to the gravity. Indeed, some of the proposed algorithms use also the angle (θ_x) between the vectors x and g to find the orientation of the body. The angle will be close to 180° when the person is standing with the arms parallel to the body. θ_x changes to values near 90° when the sensor is parallel to the floor, i.e. the same position of the body after a fall.



(a) (b) (c)

Figure 5.11.: System set-up. Position and orientation of the sensors (a), angle measured by the accelerometer (b) and skeleton joints exploited by the algorithms (c).

It worth noting that the SHIMMER device measures both the gravity acceleration and the body motion. These two components are added together in the raw data provided in output by the sensor. If the first one is useful to find the orientation of the device, the second component must be deleted, otherwise the reliability of the angle θ_x can be altered. Therefore, before using the first formula in Eq. (2.18), the body acceleration must be separated from the useful information. As proposed by Khusainov et al. in [165], the ADL performed by a person have a frequency in the interval 0.3-3.5 Hz. In consideration of that, it is possible to delete the body acceleration components using a Butterworth filter of the third order with a cut-off frequency of 0.5 Hz. The filter will be applied to the acceleration components (X, Y, Z) before finding the angle θ_x.

In Figure 5.11c is shown the human silhouette in front view with superimposed 14 out of 25 3D joints provided by the skeleton engine. They are sketched with blue dots and connections are highlighted with blue lines. The red joints are called $SPINE\_BASE$ (J_SPB) and $SPINE\_MID$ (J_SPM) [118] and will be used by the proposed algorithms. The first point refers to the base of

the spine in the pelvis, while the second represents the middle point of the trunk.

5.1.3.1. Algorithm 1-2

The first algorithm uses the data provided by the accelerometer placed in the wrist and by the depth sensor (skeleton). The algorithm finds the following features:

- displacements of the skeleton joints;
- acceleration amplitude (M_{acc});
- orientation angle of the wrist (θ_x).

The algorithm uses the y-coordinate of the 3D skeleton joint $SPINE\_BASE$ (J^y_{SPB}) in order to find vertical movements of the body that can be related to a fall. The variable $J_{SPB}(t)$ will be used to refer the position of the joint during time. For each frame, $J^y_{SPB}(t)$ is compared with the reference value $\overline{J^y_{SPB}}$, set as the first useful y-coordinate recorded during the starting phase of the test. In particular, the first time that the person is recognized by the camera, the system saves the coordinates of the joint J_{SPB} in $\overline{J_{SPB}}$.

When the difference (J_{diff}) between $J^y_{SPB}(t)$ and $\overline{J^y_{SPB}}$ is greater than 50 cm (th_{sk}), the algorithm evaluates the information provided by the other sensor. In particular, the system saves the time when J_{diff} exceeds the threshold th_{sk}, this instant is called ($\overline{t_{ir}}$). Using the time synchronization procedure already described in Section 3.1.3, the acceleration values are mapped in the reference time of the depth camera. Then, the acceleration magnitude is analysed for a time interval of 2 s centred in $\overline{t_{ir}}$. If an acceleration peak of a 3 g (th_{acc}) is found, the algorithm starts the third check. Indeed, when a person falls the body undergoes to an acceleration that ends with an impact to the floor. The accelerometer shows this trend as a peak in the monitored signal.

In the third control, the system will assess if the person is lying on the floor after the fall. Consequently, the orientation of the sensor must be close to 90° (±20°) for at least 0.5 s in the next 4 s after $\overline{t_{ir}}$. When all the previous three conditions are verified, the system triggers a fall alarm.

Algorithm 2 implements the same controls previously described. The only difference is the configuration of the wearable sensor that is now placed in the belt. The Pseudocode 5.2 summarizes the conditions that must be verified to send an alarm.

5.1.3.2. Algorithm 3

The third algorithm does not use the orientation of the wearable device, but exploits the following information:

- displacements of the skeleton joints;
- distance gap between $J_{SPB}(t)$ and the floor;
- acceleration amplitude (M_{acc}).

Algorithm 5.2 Description of algorithms 1-2.

Input: $\left(\overline{J_{SPB}}, J_{SPB}(t), M_{acc}(t), \theta_x(t)\right)$

1: **while** $\overline{J^y_{SPB}} - J^y_{SPB}(t) < th_{sk}$ **do**
2: normal activity, test next skeleton
3: **end while**
4: $\overline{t_{ir}}$ =instant of irregular activity
5: **if** $max\left(M_{acc}\left[\overline{t_{ir}} - 1s, \overline{t_{ir}} + 1s\right]\right) > th_{acc}$ **then**
6: **if** $\theta_x\left(\overline{t_{ir}}, \overline{t_{ir}} + 4s\right) == 90° \pm 20°$ at least for $0.5s$ **then**
7: fall detected
8: **end if**
9: **end if**

As implemented in the previous two algorithms, also in this case fast variations of $J^y_{SPB}(t)$ are monitored. To evaluate the distance between $J_{SPB}(t)$ and the floor, it is necessary to model the surface of the floor as a plane. The equation that easily represents this element is:

$$ax + by + cz + d_{cst} = 0 \qquad\qquad v_n = [a, b, c] \tag{5.11}$$

where the constant d_{cst} is calculated from:

$$d_{cst} = -(ax_0 + by_0 + cz_0) \tag{5.12}$$

v_n is the normal vector to the plane and $P_0 = [x_0, y_0, z_0]$ is a point on the surface. In this specific case, P_0 is set as the foot joint of the first useful skeleton. The unit vector v_n is calculated knowing the position of the joints J_{SPB} and J_{SPM}, as well as supposing that the person is in vertical position with respect to the floor when the algorithm starts to set the value of v_n.

The following equation is used to find the unit vector v_n:

$$v_n = \frac{\overline{J_{SPM}} - \overline{J_{SPB}}}{\|\overline{J_{SPM}} - \overline{J_{SPB}}\|} \tag{5.13}$$

The distance to monitor is:

$$dist_{SPB}(t) = \frac{|v_n \cdot J_{SPB}(t) + d_{cst}|}{\|v_n\|} \tag{5.14}$$

When $dist_{SPB}(t)$ is lower than a specific value ($th_{flr} = 20$ cm), the algorithm starts to evaluate the trend of $M_{acc}(t)$, in an interval of 2 s, to find an acceleration peak greater than th_{acc}. If the peak is found, the algorithm triggers an alarm.

The Pseudocode 5.3 describes the fundamental steps implemented by algorithm 3.

Algorithm 5.3 Description of algorithm 3.

Input: $\left(\overline{J_{SPB}}, J_{SPB}(t), M_{acc}(t)\right)$

1: **while** $\overline{J^y_{SPB}} - J^y_{SPB}(t) < th_{sk}$ **do**
2: normal activity, test next skeleton
3: **end while**
4: $\overline{t_{ir}}$ =instant of irregular activity
5: **if** $dist_{SPB} < th_{flr}$ **then**
6: **if** $max\left(M_{acc}\left([\overline{t_{ir}} - 1s, \overline{t_{ir}} + 1s]\right)\right) > th_{acc}$ **then**
7: fall detected
8: **end if**
9: **end if**

The described algorithms are tested with a dataset ($Dts\_ADLFall$) of 11 different people aged between 22-39 and height in the interval 1.62-1.97 m. Each person has been performed, in a laboratory environment, three repetitions of the activities described in Table 5.4. The implemented ADL and falls have been chosen from the most common group of activities described in the studied publications [28, 35]. During the ADL, no restrictions are imposed and people are free to perform the movements as natural as possible.

During the recording of the different falls, a mattress, with a thickness of 15 cm, has been used to avoid that the volunteers involved in the test would have negative consequences. After the fall, people are invited to stay in lying position in order to correctly simulate a real fall. The fall that ends up sitting is an exception, indeed in this case the shoulder of the person must be in contact with a vertical surface such as a wall or furniture.

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 5.4.: ADL and falls performed during the tests.

Also in this case, the dataset is online [122] and the published streams are:

– raw acceleration data from each axis of the accelerometers;

– skeleton joints;

– depth stream;

– timestamps related to the previous data.

The depth data is not used as input to the algorithms, but it is another useful stream that can be exploited in the future.

The proposed algorithms have been implemented in Matlab language and they need a computational time close to 2-6 ms to process samples in the interval length of 2.5-1.5 s. Consequently, the complexity of the proposed solutions is very low.

The performances are evaluated in terms of sensitivity (Sen), specificity ($Spec$) and accuracy (Acc).

$$\begin{aligned} Sen = \frac{TP}{TP+FN} \cdot 100 \quad Spec = \frac{TN}{TN+FN} \cdot 100 \quad Acc = \frac{TN+TP}{TN+FP+FN+TN} \cdot 100 \\ where: \quad \begin{aligned} TP &= true\,positive\,(detected\,fall) \\ FN &= false\,negative\,(not\,detected\,fall) \\ FP &= false\,positive\,(ADL\,activity\,labeled\,as\,fall) \\ TN &= true\,negative\,(ADL\,activity\,correctly\,identified) \end{aligned} \end{aligned} \tag{5.15}$$

The results are visible in Table 5.5, where the previous three parameters are calculated for each algorithm using the $Dts\_ADLFall$ dataset. Table 5.6 shows the accuracy values calculated for each single test. Algorithm 1 is the less invasive, but also the less efficient, as denoted by its sensitivity. Analysing the accuracy results in the first column of Table 5.6, it is evident that algorithm 1 is not able to detect efficiently different falls. The problem is the angle provided by the accelerometer that sometimes, after a fall, is not close to 90°, especially in the tests labelled *Front* and *Side*. This is a consequence of a wrong wrist orientation when the person is lying on the floor since, most of the time, the sensor is not parallel to the surface.

Algorithm 2 overcomes this limitation thanks to the more accurate angle information calculated from the data provided by SHIMMER placed on the belt. In Table 5.5 all the three parameters have better values than the results of algorithm 1. However, the second column in Table 5.6 shows that algorithm 2 has a very low accuracy in the last test (18 %). In this case, after the fall, the person is not lying on the floor, but for example, his back is parallel to a wall. Therefore, the angle between the torso and the floor is not enough to verify the condition (p.6) in Pseudocode 5.2.

The best performances are reached by algorithm 3, where the distance between $J_{SPB}(t)$ and the floor is monitored instead of the accelerometer's orientation. The algorithm fails to detect a fall only in one *Front* fall test. The problem is due to the official joint estimation algorithm which provides a skeleton that does not match with the posture of the person in that specific case. Indeed, the skeleton engine has been designed to model the person movement while he is standing in front of the camera and not lying on the floor. As a consequence of this not accurate tracking,

algorithm 3 does not pass the conditions on the controlled joints (p.1 in Pseudocode 5.3) with the corresponding classification of the activity as ADL.

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

Table 5.5.: Algorithms performances evaluated in terms of sensitivity, specificity and accuracy.

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%

Table 5.6.: Accuracy for each activity.

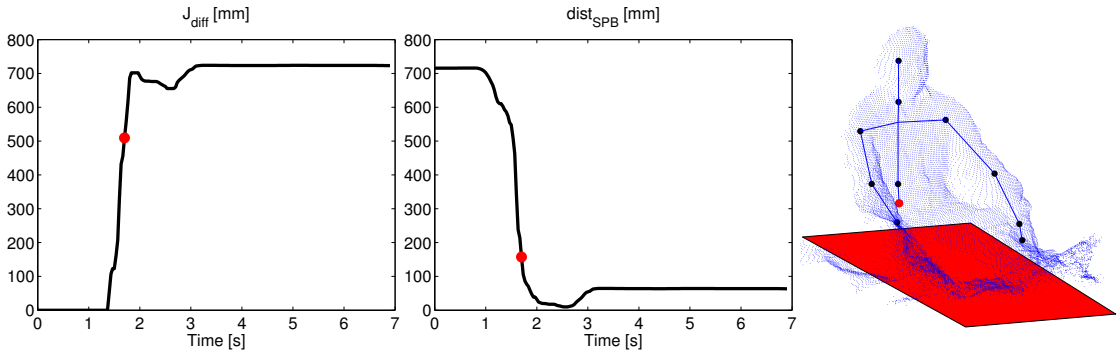
Figure 5.12a-b shows the values for J_{diff} and $dist_{SPB}$ during an *end up sitting* test. Algorithm 3 is able to correctly detect the fall because the distance $dist_{SPB}$ decreases below the threshold th_{flr} after 1.3 s from the beginning of the test.

Figure 5.12c shows the person's PC after the fall, with the floor plane (highlighted in red) used to calculate the distance $dist_{SPB}$. The skeleton is superimposed to the PC and the joint J_{SPB} is coloured in red.

To confirm the abnormal activity, algorithm 3 selects a time windows of 2 s centred in 1.8 s in the magnitude acceleration trend. Figure 5.13a shows this information, with the time window sketched with the two dashed lines. Inside this interval is visible the acceleration peak, detected by both the wearable devices.

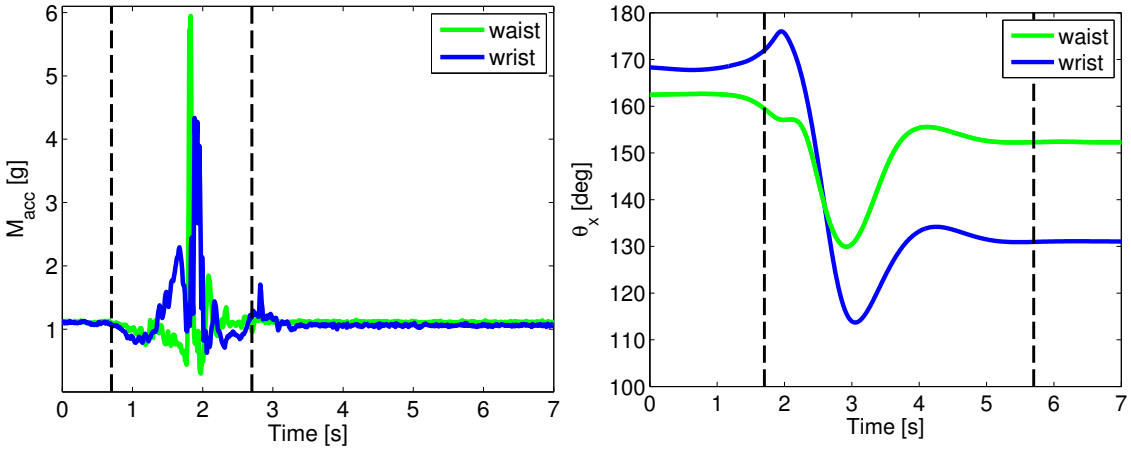
For the same test, in Figure 5.13b is visible the trend of the angle θ_x for both the accelerometers.

The value is always higher than 110° because the person never lies down to the floor. For this reason, algorithms 1 and 2 are not able to find the fall, while algorithm 3, using the distance $dist_{SPB}$ as alternative feature, is able to detect the risk situation.



(a) (b) (c)

Figure 5.12.: Trend for J_{diff} (a) and $dist_{SPB}$ (b). Person's PC with highlighted in red the estimated floor plane (c).



(a) (b)

Figure 5.13.: M_{acc} (a) and the θ_x angle (b) calculated from the raw acceleration data provided by the SHIMMER devices.

5.2. Timed Up and Go Test

The fall risk increases further to the reduction of physical abilities in the elderly. In order to prevent a possible fall, different assessment tools have been proposed. The TUG test is one of these and it is used in the hospital to monitor the gait abilities of a patient while he is performing specific motor exercises. The test starts with the person sat on an armless chair, in order to avoid that the subject uses the object during the sit-to-stand phase. When the doctor gives the start signal, the subject stands up from the chair, walks for 3 metres, turns around and finally goes back

to the chair.

Different mobility indicators can be evaluated. For example, during the sit-to-stand phase, if the patient takes a noticeable time interval to complete the movement, it is a sign of abnormality. As well as if he needs help to get up. The torso's inclination in the initial part of the test, as confirmed by Papa et al. in [166], is another relevant parameter to monitor. Indeed, information of the rigidity of the body could be inferred from the angle measurement. A hesitation time before start walking is another sign of abnormality. When the person is walking other parameters are evaluated: if he has a natural step, if the feet are dragged on the ground, if the followed direction is a straight line or not as well as the movements of the arms. In the turning phase, the number of the steps and the time necessary to complete the task are other two parameters to be deeply monitored. Finally, in the stand-to-sit phase, the considerations for the sit-to-stand part continue to be valid.

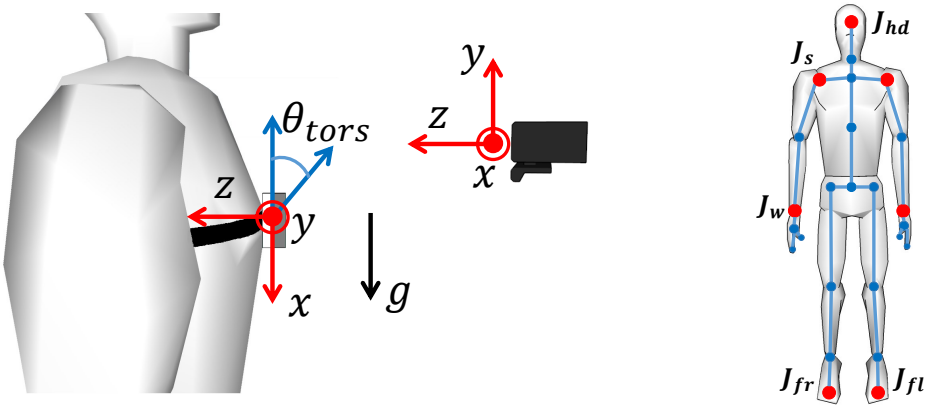
Today, the medical staff manually notes the score of the TUG test, while the patient is performing it. A limitation of this approach is that the results are strongly related to the experience of the doctor who is assessing the patient.

The idea is to devise a system that, starting from the raw data of Kinect V2 and SHIMMER, is able to provide in output an objective analysis. Another noticeable advantage of the proposed solution is that different scores per patient can be easily stored at the end of each test. This opens up the possibility to have a chronology of the tests performed during a long period and, at the same time, allows understanding if a specific therapy enhances the elder health conditions. Finally, differently from the techniques that use expensive gait analysis laboratories [55], the sensors in the proposed solution can be used in a semi-controlled environment or directly at home.

The system set-up is visible in Figure 5.14a. In order to have a test that is repeatable, the Kinect sensor must be placed at a distance of 1.5 m from the floor, in front of the subject and slightly tilted down. The distance between the sensor and the chair is 3 m and it is a trade-off between the standard minimum distance for the TUG test, the maximum range where the skeleton is still available (4 m) and size of the longer side of a room in a domestic environment. Using a belt strap, the SHIMMER sensor must be placed to the chest of the person. This configuration allows finding the torso's inclination when the subject is getting up, i.e. the angle between the body and the opposite vector to the gravity force. In Figure 5.14b are highlighted in red the skeleton joints used by the algorithm.

Taking into account the sensors set-up and the person's parameters to monitor during the TUG test, the algorithm provides in output the following information:

- torso's angle (θ_{tors});
- time duration of the sit-to-stand phase (t_{STS});
- time duration of the turning phase (t_T);
- time duration of the stand-to-sit phase (t_S);
- stride length;



(a) (b)

Figure 5.14.: Sensors configuration in lateral view and torso's tilt angle (a). Skeleton joints exploited by the algorithm (b).

- cadence time (t_{SP});
- angular speed of the arms (ω_A);
- time duration of the test (t_{TUG}).

The torso's angle can be calculated using the information provided by the wearable device. In particular, starting from the raw acceleration data, the corresponding g value can be easily found from Eq. (2.16). The angle is equal to:

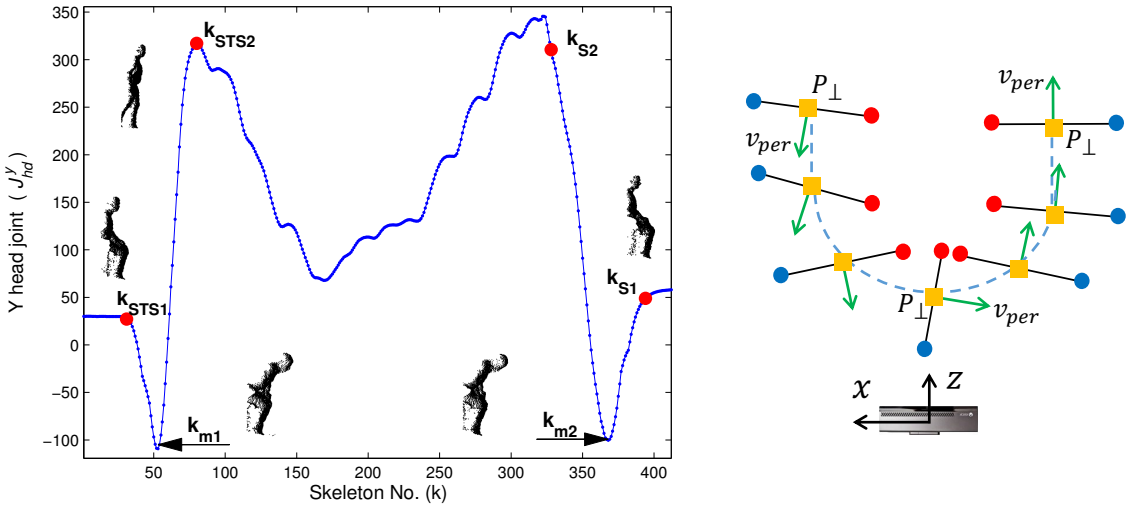
$$\theta_{tors} = max\left(90 - \tan^{-1}\left(\frac{X}{\sqrt{Y^2 + Z^2}}\right)\right) \tag{5.16}$$

The different phases of the TUG test (sit-to-stand, walking forward, turning, walking back, stand-to-sit) can be identified using the skeleton information provided by Kinect V2. In details, as visible in Figure 5.15a, k_{sts2} is the skeleton index when the subject has completed the sit-to-stand phase and he starts walking. k_{sts2} can be identified evaluating the y coordinate of the head joint (J^y_{hd}) and selecting the index when the joint reaches the maximum value. This instant verifies Eq. (5.17).

$$J^y_{hd}(k_{sts2}) = max\left(J^y_{hd}(k)\right), \quad k = 1, 2, ..., K/2 \tag{5.17}$$

Where K is the total number of skeleton for the specific test. In the same way, k_{m1} is the time, in the interval [1 k_{sts2}], when J^y_{hd} is minimum:

$$J^y_{hd}(k_{m1}) = min\left(J^y_{hd}(k)\right), \quad k = 1, 2, ..., k_{sts2} \tag{5.18}$$



(a) (b)

Figure 5.15.: y coordinate of the head during the TUG test (a). Orientation vectors of the person in the turning phase (b).

When k_{sts2} and k_{m1} are identified, it is possible to find the index k_{sts1} i.e. when the sit-to-stand phase starts. Eq. (5.19) is used to find the value.

$$|J^y_{hd}(k) - J^y_{hd}(k-1)| > T_{h1\%}, \quad k = 1, 2..., k_{m1} \qquad T_{h1\%} = |J^y_{hd}(1) - J^y_{hd}(k_{m1})|/100 \quad (5.19)$$

k_{sts1} corresponds to the first skeleton index when the difference between two consecutive J^y_{hd} is more than the threshold $T_{h1\%}$.

The calculated values are discrete indexes. To find the associated times with respect to the beginning of the test, they must be multiplied by the interval between two consecutive skeletons (Δ_t), i.e. 33 ms. Eq. (5.20) provides the time interval for the sit-to-stand phase.

$$t_{STS} = (k_{sts2} - k_{sts1})\,\Delta_t \quad (5.20)$$

When the person completes the walking phase, he starts to turn around. The orientation of the body (v_{per}) is used to find when the turning phase starts. In Figure 5.15b is visible different positions of the shoulders' joints, sketched with red and blue dots, in the horizontal plane (y is constant). The green vector (v_{per}) is parallel to the person direction and it is calculated from the head joint and its projection ($P_\perp$) in the shoulder axis (black segments). $P_\perp$ is represented with yellow squares in Figure 5.15b.

v_{per} is then compared with the z-axis of Kinect to find the orientation angle of the person (θ_{per}).

$$v_{per} = \frac{J_{hd} - P_{\perp}}{\|J_{hd} - P_{\perp}\|} \qquad \theta_{per} = cos\left(v_{per} \cdot z\right) \tag{5.21}$$

θ_{per} is close $0°(180°)$ when the person is walking in the straight line toward (from) the chair. During the turning phase the angle is close to $90°$. These different values of the angle θ_{per} are exploited by the algorithm to find the turning phase. In particular, when the person finishes to turn, the θ_{per} begins to decrease from $180°$ to $90°$. The first time that the angle passes $160°$, the algorithm marks the corresponding index as start time of the turning phase. In the same way, while the person is ending to turn, θ_{per} passes $90°$ and continues to decrease to $0°$. The turning phase is completed when θ_{per} reaches $20°$ and the time interval between the start and stop is equal to t_T.

The algorithm uses the same approach in the final part of the test, to find when the stand-to-sit phase begins (k_{s2}). In particular, when the person is in front of the chair and starts to turn for the second time, the algorithm selects the index when θ_{per} passes $20°$ to rise again.

The instant k_{m2} can be identified with the same approach used for k_{m1}, but calculated in a different interval (see Figure 5.15a):

$$J^y_{hd}(k_{m2}) = min\left(J^y_{hd}(k)\right), \quad k = k_{s2}, ..., K \tag{5.22}$$

k_{s1} will be the first index for which the following condition is true:

$$|J^y_{hd}(k) - J^y_{hd}(k-1)| > T_{h2\%}, \quad k = K, ..., k_{s2} \qquad T_{h2\%} = |J^y_{hd}(K) - J^y_{hd}(k_{m2})|/100 \tag{5.23}$$

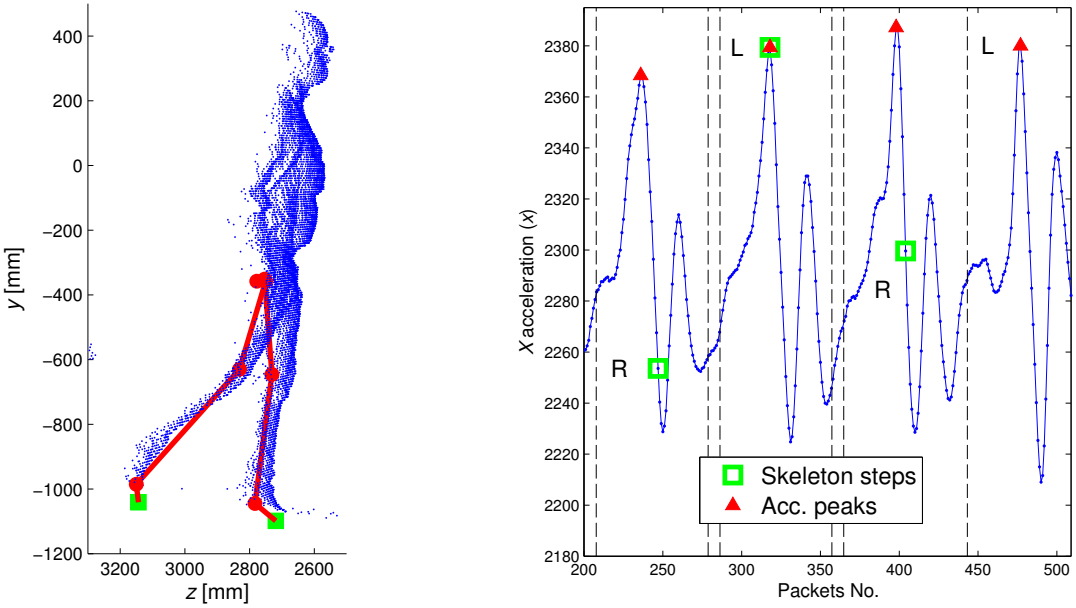
t_S can be found substituting k_{sts2} and k_{sts1} in Eq. (5.20) with k_{s1} and k_{s2}, respectively.

Now it is possible to calculate the total duration of the test, indeed t_{TUG} is equal to the time interval between the indexes k_{sts1} and k_{s1}. The equivalent time can be calculated from these two skeleton indexes using again Eq. (5.20).

Once all the different phases of the test are identified, the algorithm finds the cadence and the angular velocity of the arms while the person is walking in straight line. Initially, the skeleton data is used to approximately find time instants of the steps. To this end, the z coordinate of the left (J^z_{fl}) and right (J^z_{fr}) feet are compared for each skeleton and the difference between these two points is calculated. The step is equal to the maximum values of this difference.

Figure 5.16a shows the PC of a volunteer, obtained from the depth frame of the person inverting Eq. (2.4), with superimposed the lower part of the skeleton coloured in red. In this specific case the step, calculated from the position of the feet joints, is equal to 40 cm. Figure 5.16b is the acceleration in the X-axis provided by SHIMMER for the interval that begins with the initial phase of the test until the turning one.

Thanks to the synchronization solution, the step instants previously found can be mapped in the



(a) (b)

Figure 5.16.: PC of the person with superimposed the lower part of the skeleton and the feet joints (a). Acceleration values, provided by the SHIMMER device, in the X-axis (b).

acceleration domain. With reference to the case in Figure 5.16a, the corresponding value in the SHIMMER domain is the first green square in Figure 5.16b. Now the algorithm tries to improve this result finding the peak in an interval close to the green square. The interval, indicated with vertical dashed lines, is centred in the initial value (green square) and its width is equal to the mean time duration of all steps. The precise step instants are highlighted in Figure 5.16b using red triangles and will be used to find the cadence.

Due to Kinect set-up, when the person is close to the camera (turning phase), it is possible that the feet joints are not visible and the skeleton information could not be used to find the last step. However, the system can still find the step using only the acceleration information. Indeed, as visible in the final part of Figure 5.16b, the system found a peak because it has a height comparable with previous ones.

When the peak is found in the acceleration domain, it must be linked to the correct foot. To this end, the algorithm goes back to the skeleton domain and, for each acceleration peak, finds the foot joint that is close to the sensor in that specific time. As visible in Figure 5.16b, the peak is marked with label R or L if it is respectively generated by the right or the left foot. For the last peak, the label is set as the opposite letter selected for the previous foot.

After the turning phase, when the person is walking back to the chair, it is not possible to use the skeleton data to find the initial values of the steps. The problem is that the accuracy whereby the skeleton represents the person is too low and, as consequence, the joints positions of the feet

are not reliable. In this case, the acceleration data is the only information used to find the steps. The algorithm exploits the peak characteristics of the first walking phase to identify similar values in the second one.

Starting from the step indexes, the algorithm finds the cadence time, as the time interval between two peaks generated by the same foot. Whereas, the stride length is the distance covered by the foot in the same interval.

The indexes of the cadence are used to set the intervals in which the angular velocity of the arm is calculated. If $J_s(k) = [J_{sx}(k)J_{sy}(k)J_{sz}(k)]$ is the coordinate of the shoulder joint at k^{th} skeleton, and $J_w(k) = [J_{wx}(k)J_{wy}(k)J_{wz}(k)]$ is the wrist coordinate for the same arm, the angular velocity is:

$$w_A(k) = \frac{1}{\Delta_t} cos^{-1}\left(\frac{J_{sw(k)}}{\|J_{sw}(k)\|} \cdot \frac{J_{sw}(k-1)}{\|J_{sw}(k-1)\|}\right) \qquad \begin{matrix} J_{sw}(k) = J_w(k) - J_s(k) \\ J_{sw}(k-1) = J_w(k-1) - J_s(k) \end{matrix} \tag{5.24}$$

$J_{sw}(k)$ is the vector that starts from the shoulder and ends in the wrist, while $J_{sw}(k-1)$ is another vector that starts from the same shoulder joint, but ends in the coordinate of the wrist that belongs to the previous skeleton. The angle between these two vectors divided by the frame rate is equal to the angular velocity for the k^{th} skeleton.

The test has been performed by 20 people aged between 22-39, having different weights and heights. Each subject completes the test three times for a total of 60 different repetitions. The dataset (Dts_TUG) is composed of the following elements:

- the skeleton coordinates;
- the acceleration data;
- depth frames;
- timestamp related to each sample.

The depth stream has not been exploited by the proposed algorithm, but can be useful for future projects or interesting for other research groups that need this type of information.

Table 5.7 highlights the fundamental parameters calculated by the algorithm over the dataset. Taking into account that the tests have been performed by healthy people, the results are quite similar, as confirmed by the low values of standard deviation when compared with the corresponding mean values. The average time to perform the test is 10 s while the sit-to-stand phase is performed in about 1.5 s. The stand-to-sit phase is longer than the sit-to-stand phase because in the interval is counted also the turning phase. The angular velocity is calculated for the left arm during the first cadence.

Parameters	Mean value	Minimum	Maximum
θ_{tors} [deg]	45.3 ± 13.53	18.0	64
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 [cm]	956 ± 147	743	1295
ω_A [deg/s]	107 ± 21	67	158

Table 5.7.: Parameters in output from the algorithm.

The torso's angles have been compared with the equivalent angles obtained evaluating manually the depth streams of each single test. The differences are negligible and this confirms the reliability of by the angle calculated from the accelerometer.

5.3. Tracking Solutions for Intake Analysis

This section describes another application designed for AAL. The idea is to track automatically the movements performed by an older person during a meal, in order to assess his intake actions and alert the informal care or relatives when the behaviours are not compliant with a correct diet.

In particular, the depth stream from Kinect V1 is used as input data, in the same configuration described for the fall detection algorithm (Section 5.1). The sensor, installed on the ceiling at a distance of 3 m from the floor, monitors an area where usually the person has the main meal such as the kitchen or the living room.

As illustrated in the next subsection, the depth frame is elaborated to find the person's PC. The next step is monitoring the movements performed by the subject. Therefore, the self-organizing algorithms, introduced in Chapter 4, are exploited to fulfil this step. The SOM and SOM_Ex algorithms fit an initial model to the person's PC, whereas the GNG will find itself the better model to the manifold in input. For each algorithm, the characteristic parameters are properly tuned considering the computational requirements and the size of the manifold. The coordinates of the most important nodes are then compared with a ground truth to find the technique that better approximates the movements.

Afterwards, the table pixels in depth frame are mapped in the RGB frame using the spatial synchronization procedure described in Section 3.2. The colour information is then exploited to find useful objects (plates, glasses) on the table.

In the final part of this section, a direct data fusion example is described. The drinking action is inferred combining the tracking solution and the position of a glass in the RGB frame.

Two datasets are used with the algorithms described in the next sections. The first one (*Dts_in-*

take_1) stores 48 tests where 35 volunteers, aged between 22-38 years old and height in 1.62-1.97 m, perform eating and drinking actions. In particular for each test, the person enters in the monitored area and sits near the table, while Kinect V1 records the depth (320×240) and RGB (640×480) streams. He starts eating and drinking using cutlery and glass placed on the table.

The second dataset (*Dts_intake_2*) is similar to the first one, but now 20 people, aged between 23-41 years old and height in 1.62-1.93 m, have been recruited for a total of 60 tests with 3 repetitions per person. The recorded streams are depth (320×240) and IR (640×480) and the protocol for each volunteer is the following:

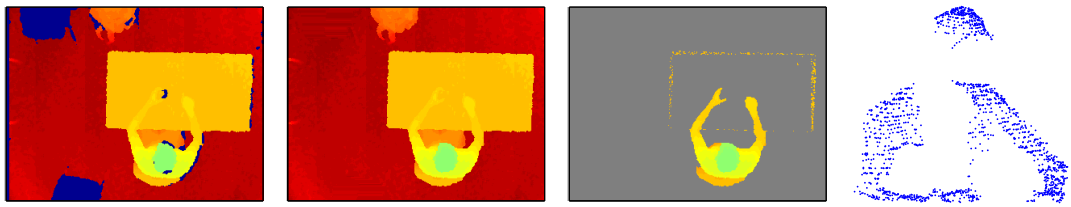
- repetition 1: eating a snack using the hand and drinking water from a glass;
- repetition 2: eating a soup with a spoon and pouring/drinking water;
- repetition 3: using knife and fork for the main meal and finally wiping the mouth with a napkin.

Each volunteer wears three IR markers; one on the head and two on the hands. They are used to have a ground truth that will be compared with the output of the self-organizing algorithms.

5.3.1. Input Values to the Self-Organizing Algorithms

Figure 5.17 shows the processing on a typical depth frame to provide in output the person's PC. Starting from the raw data (Figure 5.17a), the "holes" in the image are filled with the same solution used in Section 5.1.1.1 (Figure 5.17b).

A reference depth map, framed before that the person enters in the scene, is compared with the processed depth frame. The output is visible in Figure 5.17c, where the foreground pixels have a colour different from grey. A median filter with a window of 5×5 pixels is applied to the foreground frame to remove the noise elements. In particular, in Figure 5.17c, they are the pixels at the border of the table. Here the distance information has more variability compared with other areas such as the centre of the table or the floor and for this reason these pixels can be erroneously marked as foreground.



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

Figure 5.17.: Processing of the depth frame. Raw data (a), zero-filling operation (b), foreground pixels (c) and person's PC (d).

The same labelling algorithm described in Section 5.1.1.2 can be exploited to find now the biggest blob in the monitored area, with the assumption that it coincides with the person's blob. Finally, the intrinsic depth camera parameters are exploited to calculate the person's PC (Figure 5.17d).

Each point in the PC will be randomly selected as input value to the self-organizing algorithms. Tests performed on the dataset *Dts_intake_*1 provide a mean value of 17 K 3D input points per frame, with a maximum/minimum value of respectively 9 K and 25 K.

In addition to the person's PC, the three algorithms also need the orientation of the person. Indeed, when the datasets have been built, no restrictions were imposed to the volunteers and they were free to choose whatever side of the table. This information is exploited by the SOM/SOM_Ex solutions to properly rotate the initial model before starting the fitting process. On the other hand, after the adaptation phase, the GNG will use this information in a custom post-processing algorithm to find the nodes that model the hands. Motivated by this fact, Figure 5.18a shows a depth frame where the person is sitting close to the table. The vector vr_{std} indicates the standard direction and it is parallel to the v-axis, while vr_b is the body's direction.

The system uses the person identification solution, introduced in Section 5.1.1.3, to verify that the blob is a person. Then, it finds in the person's PC the point that models the head (J_{hd}) as the mean of the closest points to the sensor. J_{hd}, using Eq. (2.4), is mapped in the depth frame and the 2D point is indicated in Figure 5.18a by J^f_{hd}.

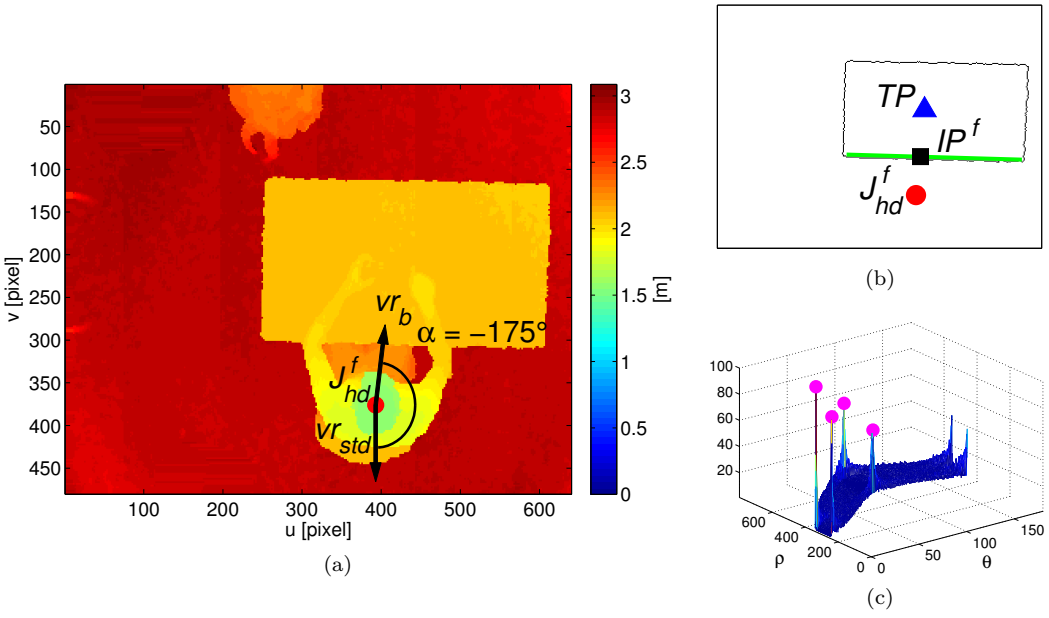


Figure 5.18.: Depth frame with the person's direction (vr_b) (a). Table's border (F_{cnt}) and points used to find the body direction (b). Table representation in the Hough domain (c), the four peaks represent the table's borders.

To find vr_b, the assumption is that the vector starts from the J^f_{hd} and is perpendicular to the closest border of the table. The object is identified on the reference frame when the system is activated for the first time. The program checks that a table is inside the monitored area using the RANSAC algorithm [167]. Indeed, due to the sensor set-up, the system exploits the properties of the table as it is a flat surface with uniform depth values. Subsequently, the program, starting from the binary frame (F_{tab}) where only the pixels of the table are set to 1, finds the borders of the object (F_{cnt}) using the erosion operation (A.1) with a 2×2 matrix as kernel.

$$F_{cnt} = F_{tab} - (F_{tab} \ominus kern) \tag{5.25}$$

It is worth noting that the result is very similar to the output of an edge detection algorithm, but in this case, it is not necessary to set up the correct threshold. The frame F_{cnt}, where only the borders of the table are set to 1, is visible in Figure 5.18b. The frame is given in input to the Hough transform (A.3) to find the angles (θ) and radius (ρ) of each straight line that models the table's border. Figure 5.18c shows the equivalent representation of the table's borders in the Hough domain (accumulator plane). The four peaks, highlighted by magenta dots, are the couples (θ, ρ) for each border.

For each couple, it is calculated the projection point of J_{hd} to the line. In particular, in Figure 5.18b the green line is the border to find, J^f_{hd} is the joint of the head and IP^f is the intersection point. The third point, called Test point (TP), is an extension of the direction that starts from J^f_{hd} to IP^f. The body's direction is found when TP is placed inside the table, indeed in all the other three situations the point is outside the object.

Algorithm 5.4 Code to find the TP.

Input: $(\theta_{peak}, \rho_{peak}), J_{hd}$
Output: TP

1: **for all** $(\theta_{peak}, \rho_{peak})$ **do**
2: straight line general form:
3: $a = -\dfrac{cos(\theta_{peak})}{sin(\rho_{peak})},\ b = -1,\ c = \dfrac{\rho_{peak}}{sin(\theta_{peak})}$
4: $IP^{fu} = \dfrac{b(bJ^{fu}_{hd} - aJ^{fv}_{hd}) - ac}{a^2 + b^2}$
5: $IP^{fv} = \dfrac{a(-bJ^{fu}_{hd} + aJ^{fv}_{hd}) - bc}{a^2 + b^2}$
6: TP is calculated from (J^f_{hd}, IP^f)
7: **if** $TP \in table$ **then**
8: **return** TP
9: **end if**
10: **end for**

The Pseudocode 5.4 provides in output the point TP, using as input J_{hd}^{f} and the four couples $(\theta_{peak}, \rho_{peak})$. The vector that connects the points J_{hd}^{f} and TP is the vector vr_b. The angle formed by vr_{std} and vr_b is the orientation angle of the body (α) and, in this case (Figure 5.18a), it is equal to -175°.

5.3.2. Models Fitting

In this section, each self-organizing algorithm is used to fit a model to the person's PC. They have been implemented in the Matlab language and, to test the real-time performance, the scripts have been converted in C++ language using the cross-compilation tool provided by the Matlab IDE. The results are C++ files that can be directly called in Matlab environment using the Mex-file grabber function.

5.3.2.1. SOM

The proposed SOM algorithm, compared to the original description in Section 4.1, has been modified to improve its fitting properties. In particular:

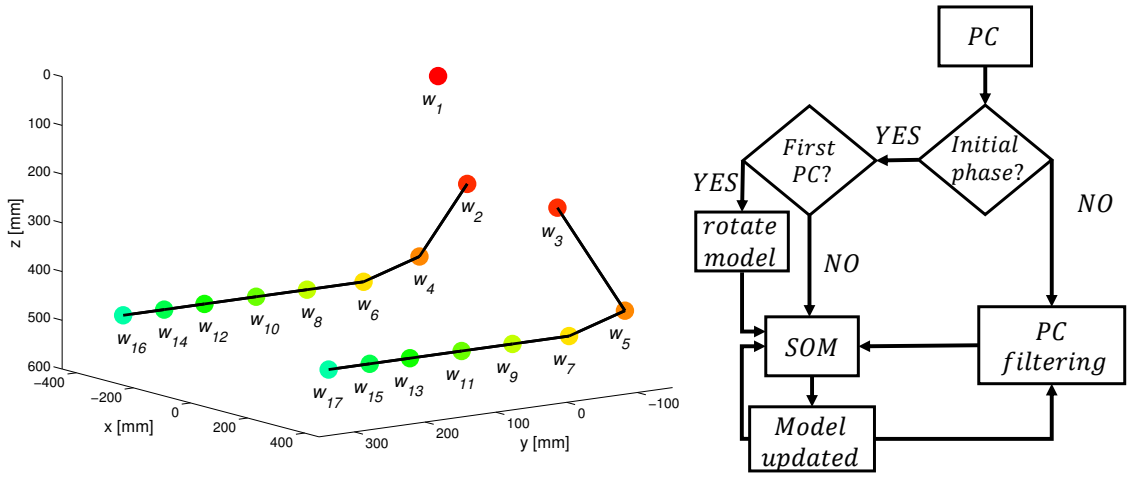
- the lattice scheme is substituted by a specific model that is more similar to the person's shape;
- the *neighbourhood function* is simplified to reduce the operations executed for each iteration;
- the manifold is given in input to the SOM more than one time (epoch).

The initial model in input to the SOM is visible in Figure 5.19a and it is characterized by N=17 elements in the 3D space. It takes the same pose of a person sat on a chair with the arms outstretched on the table. The black lines, as already introduced for the lattice model in Figure 4.1, highlight the connections between nodes. Two node chains model the arms, while a single node (w_1) is used for the head (J_{hd}). The end nodes in the chain (w_{17},w_{16}) are called J_{hl} and J_{hr} respectively to indicate the left and the right hand. Each node, except for J_{hd}, has a neighbour (w_{nbr}).

Differently from other implementations of the SOM algorithm [149, 168, 169], where the camera is in the standard frontal view and it is able to capture the entire connected parts of the body, in this situation the head PC is far away from the trunk. For this reason, J_{hd} has been separated from the other parts of the model.

The number of nodes has been selected after some testing where this parameter has tuned. Few nodes do not completely model the entire arms, but only some portions. It means that J_{hl} and J_{hr} are placed in the forearms rather than in the palms. On the other hand, more nodes do not considerably improve the tracking performance of the system.

No further nodes are used to model the chest, because it is not always present in all person's PCs in input to the SOM. Therefore, before using the SOM, an ad-hoc filtering solution is used to discard these points. As summarized in Figure 5.19b, if the PC is one of the first manifolds in



(a) (b)

Figure 5.19.: Initial model in input to the SOM algorithm (a). General scheme of the algorithm (b).

input to the SOM, then it is not filtered. Otherwise, after a training phase, the PC is modified before using the fitting algorithm.

The first time that the algorithm is used, J_{hd} is placed close to the highest points of the PC and the other nodes are rotated in the same person's direction:

$$\begin{aligned} w_{jx}^{rot} &= cos(\alpha)\left(w_{jx} - J_{hd\,x}\right) - sin(\alpha)\left(w_{jy} - J_{hd\,y}\right) + J_{hd\,x} \\ w_{jy}^{rot} &= sin(\alpha)\left(w_{jx} - J_{hd\,x}\right) - cos(\alpha)\left(w_{jy} - J_{hd\,y}\right) + J_{hd\,y} \\ w_{jz}^{rot} &= w_{jz} \end{aligned} \tag{5.26}$$

The model has the same orientation of the person and now the SOM algorithm is used to adapt it to the PC. In particular, the competition phase is the same introduced in Section 4.1.1 and the update equation for the reference vectors is as follows:

$$\begin{aligned} \Delta w_j &= \varepsilon(p) h_{BMU,j}(\xi - w_j) \\ j &: indexes\ \ of\ active\ nodes \end{aligned} \qquad \varepsilon(p) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{p/p_{max}} \tag{5.27}$$

The most important difference with Eq. (4.5) is the *neighbourhood function*. Here $h_{BMU,j}$ does not depend on time (p), but is set equal to 1 for the BMU, 0.5 for its neighbour nodes, and 0 for all the other nodes. The neighbour nodes of the BMU are the first nodes connected by a line in Figure 5.19a. For example, if the BMU is the node w_7, the neighbour nodes that will be updated are w_5 and w_9. Thanks to this simplification, the number of operations to execute for each iteration are reduced.

The *learning rate function* $\varepsilon(p)$ has been selected as the solution proposed in other similar works that use the SOM as fitting algorithm [149, 168]. ε_i and ε_f are respectively the initial and final learning rate. Since the ratio $\varepsilon_f/\varepsilon_i$ is less than 1, the *learning rate function* decreases with the passing of the time. p is the index of the point in the PC and p_{max} is the total number of points used as stimuli. The PC will be shown 10 times to the SOM and, when all the points (L) in the PC are evaluated, a so-called epoch ends. In this way, the adaptation of the model is reached just in the first processed frames.

Since no constraints are imposed to the position of J_{hl} and J_{hr}, when a hand is close to the face is possible that the head PC attracts too much the node. As a consequence, the joint remains in the head even after the arm moves in another area. Eq. (5.28) forces the joint J_{hr} to stay close to its neighbour when the distance between these nodes exceeds a threshold (γ) of 250 mm, derived from experimental tests. A similar equation is used for the J_{hl}.

$$J_{hr} = w_{nbr} + \gamma \frac{J_{hr} - w_{nbr}}{\|J_{hr} - w_{nbr}\|} \tag{5.28}$$

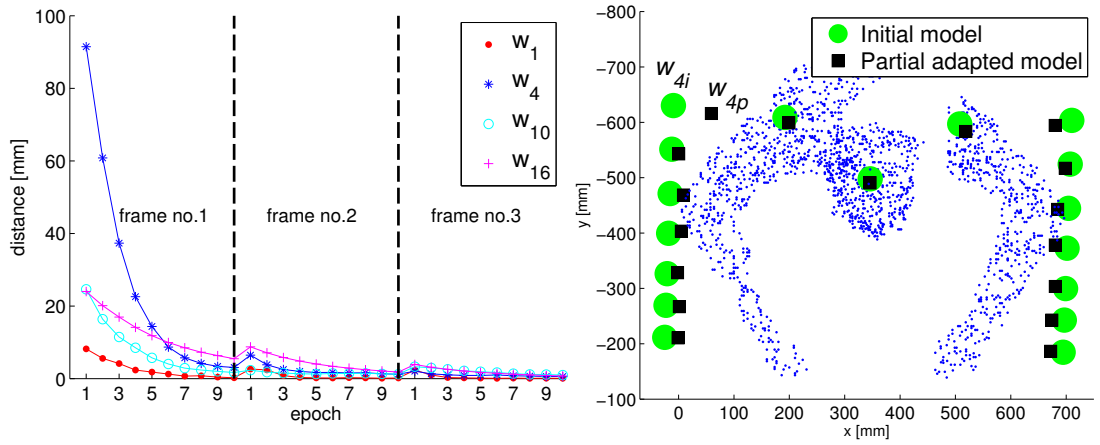
When a new PC, next to the first one, is calculated by the system, the model in input to the SOM will not be the initial network visible in Figure 5.19a. Indeed, the system will use the network trained in the current PC as the new start point for the next process. In this way, the changes in the displacement of the nodes are very small, because differences between two consecutive PCs are negligible.

Table 5.8 summarizes the parameters used by SOM algorithm. To justify this configuration, Figure 5.20a shows the displacement in time of four nodes (w_1, w_4, w_{10}, w_{16}). In particular, three consecutive PCs are provided in input to the SOM and, at the end of each epoch, a partial adapted model is obtained. To quantify how fast the fitting of each node is, position displacements are calculated between consecutive epochs. For instance, in the first epoch of the frame no.1, the displacement for the node w_4 is equal to 90 mm. In Figure 5.20b is visible the person's PC in the x-y plane (top view). The green nodes are the initial model after the use of Eq. (5.26), while the square elements represent the partial adapted model after the first epoch of the frame no.1. The difference between nodes labelled as w_{4i} and w_{4p} is equal to 90 mm. In the next epochs, this displacement decreases until the 10^{th} epoch of the frame no.3. At this frame, the model is adapted to the PC and the convergence is confirmed by differences close to 1 mm.

Net. size	ε_i	ε_f	p_{max}	γ
17	0.002	0.001	10L	0.005

Table 5.8.: Selected parameter for SOM algorithm.

In the first epoch of every new frame, the monotonic trend of curves is not respected. This feature is related to the implementation of $\varepsilon(p)$ that, in this case, takes the maximum value compared to other iterations in the same frame. Consequently, the SOM adapts the model faster in the first epoch than in following ones. Figure 5.20a also tells us that, with a frame rate of 30 fps, the initial



(a) (b)

Figure 5.20.: Displacement of nodes w_1 w_4 w_{10} w_{16} in the first three frames (a). Displacement of node w_4 for the first epoch in frame no.1 (b).

model needs 90 ms to perfectly match the PC. As a consequence for the next frames, taking into account that the displacement of the person can be considered very small if compared with the frame rate, the SOM is able to update the network quite fast and always within 30 ms.

Sometimes, when the algorithm finishes processing a PC (ends of 10^{th} epoch), the nodes that model the shoulders ($w_2 - w_4 - w_6$ and $w_3 - w_5 - w_7$) could be placed in wrong positions, because attracted by the opposite shoulder's PC (Figure 5.21a). This situation must be corrected, otherwise the chain of the arm can be inverted with negative consequences for the tracking of the movements.

The system, after the SOM algorithm, finds this situation and forces these elements to shift in the correct direction indicated by the black arrows. The shift direction (v_R) is orthogonal to the model orientation. As visible in Figure 5.21b, these nodes are correctly moved in the right shoulder's PC.

This situation for triplet $w_2 - w_4 - w_6$, is detected by the algorithm when the following inequality is verified:

$$v_R \cdot (w_i - w_1) < 0, \quad \forall i \in \{2, 4, 6\} \tag{5.29}$$

The same equation can be used for the other triplet, but v_R must be rotated by 180°.

As previously introduced, after the first three frames (*initial phase* in Figure 5.19b), the model in output from the SOM has completed the fitting process to the PC (Figure 5.20). For next iterations, the chest PC must be discarded from the input manifold because it is not considered in the initial model and, as a consequence, it could attract some nodes placed in the arms. At the same time, in the PC there could be some outliers such as the back of the chair. This object

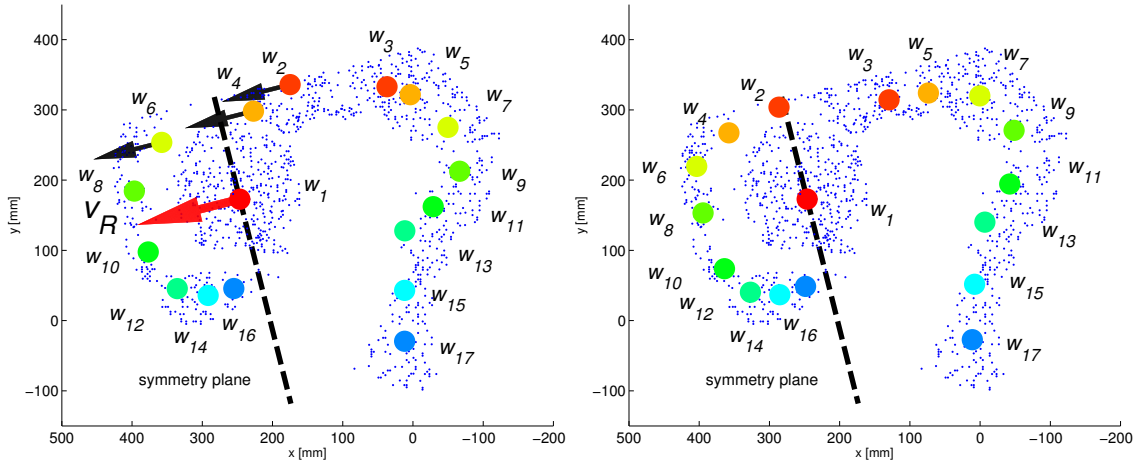


Figure 5.21.: Drift problem for the nodes w_2-w_4-w_6. The triplet moves to the left shoulder's PC (a). Result after the correction (b).

cannot be removed by the initial processing on the depth frame, described in Section 5.3.1, because it is too close to the person. If these elements (chest and chair) are not removed from the PC the model will not be able to correctly track the movements performed by the person.

An adaptable box, visible in Figure 5.22a, is exploited to select and delete unwanted parts. Its dimensions are related to the characteristics of the PC and to the distances between the model nodes after the *initial phase*. It is superimposed to the manifold using the coordinates of J_{hd}. The points are discarded if they are:

- inside the box (remove chest);
- behind the box (delete chair).

The box is represented by 7 planes, each one is identified by an application point (Q) and a vector (v) respectively indicated with red squares and magenta arrows in Figure 5.22a. They are computed from:

- body direction (vr_b) and J_{hd};
- vertical gap between J_{hd} and w_2: $z_{gap} = \|J_{hd\,z} - w_{2\,z}\|$;
- distance between w_2 and w_3: $d_{ss} = \|w_3 - w_2\|$;
- IP, corresponding 3D point of IP^f.

The distances d_{ss} and z_{gap} do not change during the execution and they are calculated using the nodes coordinates of the adapted model to the third PC.

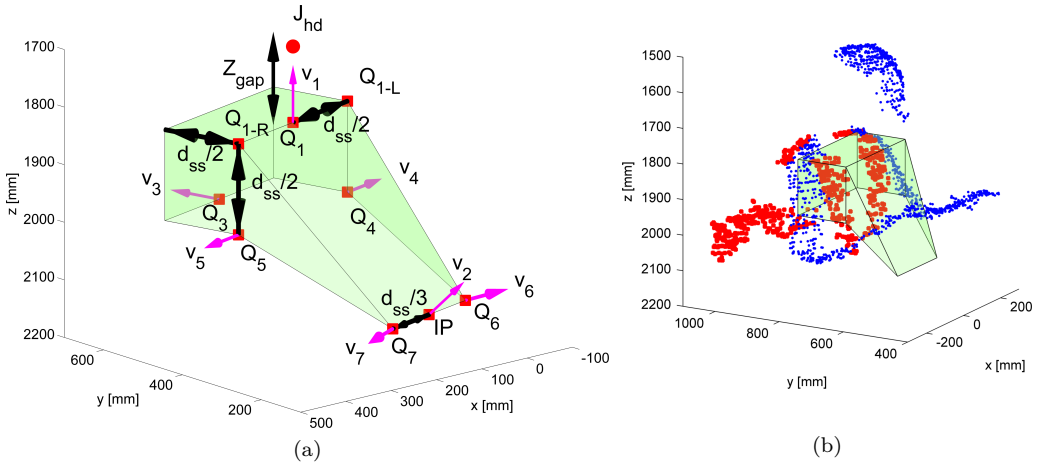


Figure 5.22.: Adaptable box (a). The chair and the chest elements are selected (b).

The Q points are found from the values in the double arrows depicted in Figure 5.22a. The vectors are calculated using the points in the border of the corresponding plane. For example, v_2 is equal to:

$$v_2 = (Q_1 - IP) \times (Q_7 - IP) \tag{5.30}$$

The same equation is used with the triplets $(Q_{1R} - Q_5 - Q_7)$ and $(Q_{1L} - Q_4 - Q_6)$ to find respectively the vector v_7 and v_6.

The points (ξ) that satisfy Eq. (5.31) are inside the box and must be deleted. Instead, Eq. (5.32) is used to find the points of the chair.

$$v_i \cdot (\xi - Q_i) < 0, \quad \forall i \in \{1, 2, 3, ..., 7\} \tag{5.31}$$

$$v_3 \cdot (\xi - Q_3) > 0 \ \cap \ (\xi_z - J_{hd\,z}) > z_{gap} \tag{5.32}$$

It is worth noting that the dimension of the box follows the changing in person's PC. Indeed, for each new PC the box dynamically shrinks or stretches its shape according to the position of J_{hd} and IP.

The result is visible in Figure 5.22b, the red points highlight the elements to discard. On the other hand, the useful points (blue) are given in input to the SOM to update the network.

5.3.2.2. SOM_Ex

This subsection presents how the SOM_Ex algorithm has been exploited to track the person's movements. The initial model in input to the SOM_Ex is shown in Figure 5.23a. The PC obtained from the depth frame, is directly used as input to the SOM_Ex without a pre-processing

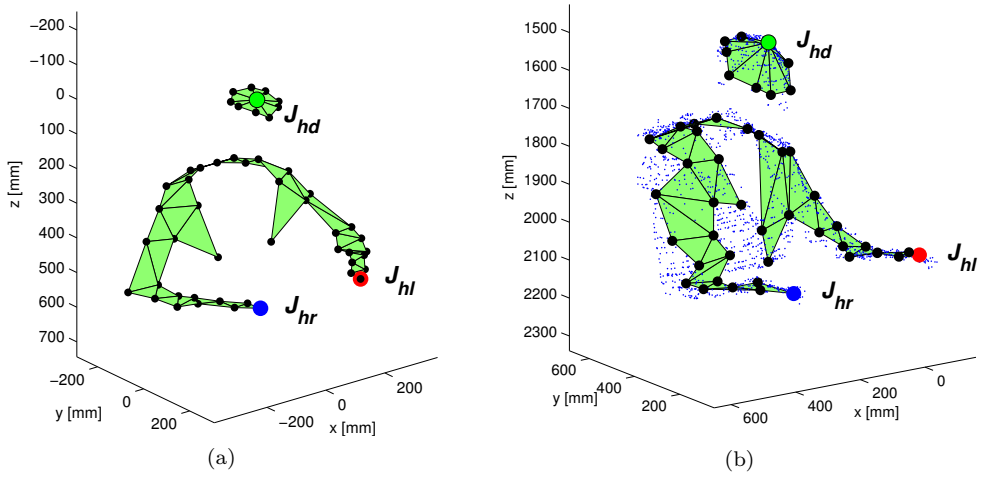


Figure 5.23.: Initial model (a) and adapted model (b) by the *SOM_Ex* algorithm.

phase.

The model is characterized by 50 nodes, for a total of 47 planes, and 9 of them are used to model the head. The head cluster is separated from the torso, as in the SOM network too, to have a more similar model to the distribution of the PC. The correct number of nodes has been tuned after some tests. The number 50 is sufficient to have an accurate tracking of the movements. Regarding the segments, defined in Section 4.2, they have not been used here because are not enough stable to represent in the correct way different positions of the arms.

A direct comparison with the network used in SOM highlights an increase of three times in number of nodes. The chains, previously used by the SOM to model the arms, are now merged in a single group. In this way, it is no longer necessary the pre-processing phase to delete the chest PC, because now it is considered in the model.

No modifications have been introduced in the original equations proposed by Coleca et al. [147] and the *learning rate function* $\varepsilon(p)$ has been chosen equal to the formula selected for the SOM (Eq. (5.27)).

Before the first use, the model must be correctly centred and rotated exploiting the same operations used for the SOM network (Eq. (5.26)). Whereas, for next PCs, the model trained in the previous frame is used as input to the SOM_Ex.

In Table 5.9 are visible the characteristic parameters used by the SOM_Ex. The PC is undersampled of a factor equal to 10, it reduces the computational time under 60 ms. ε_i and ε_f are respectively the initial and the final learning parameters used in the $\varepsilon(p)$ definition. They are two order of magnitude higher than the corresponding values in the SOM implementation. This is justified by the greater number of nodes that now have less probability to become BMU.

For each PC the SOM_Ex, as visible in Figure 5.23b, provides in output the trained model.

Finally, the three most important nodes, that will be used to compare the tracking performance of this algorithm with the SOM and GNG results, are labelled in Figure 5.23 as J_{hd}, J_{hr} and J_{hl}. The first one is the node in the centre of the head cluster, whereas the other two joints are the nodes coloured in blue and red, respectively.

Net. size	PC Step	ε_i	ε_f
50	10	0.5	0.4

Table 5.9.: Parameters used for the SOM_Ex algorithm.

5.3.2.3. GNG

The parameters used by the GNG algorithm are here analysed to find the better configuration, taking into account the application purpose.

Network Size

The right value for the total number of nodes (N) in the network could be correctly identified starting from the required precision i.e. studying the quality with which the network fits the input. Some direct measurements are possible, for example the *quantization error* (QE) proposed by Kohonen in [143] is defined as follows:

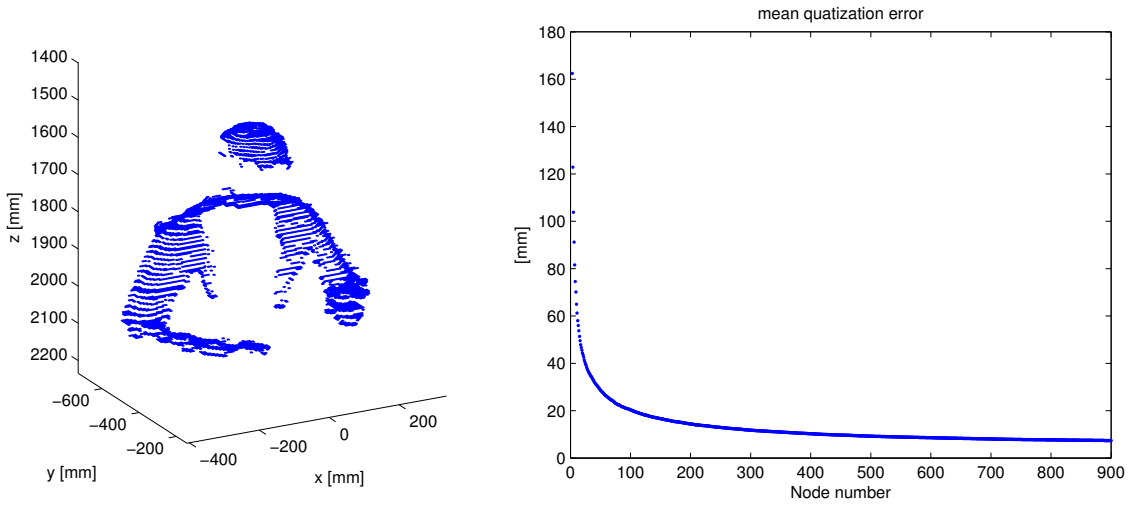
$$E = \frac{1}{L}\sum_{i=1}^{L} \|w_{s\xi} - \xi\| \tag{5.33}$$

where L is the size of the PC, $w_{s\xi}$ is the closest node to an input point (ξ). The QE is inversely proportional to N, indeed for the same self-organizing algorithm, given two trained networks where the first one has double nodes than the second, the QE will be lower in the first model. It is due to a better distribution of the nodes in the first network that covers the PC with more elements.

To extend this idea, the PC in Figure 5.24a is given in input to the GNG algorithm that will run for 900 different tests. The variables ($\varepsilon_w, \varepsilon_n, \lambda, \alpha$,etc.) in the GNG (Pseudocode 4.1) do not change, whereas N will be incremented for each test. The GNG is executed for a period considered adequate to the dimensions of the PC and when a new test starts, the network is initialized with two nodes. The QE is calculated at the end of each test and the result is shown in Figure 5.24b.

As visible in the curve, a number of 100 nodes is adequate to represent the PC since, if N increases, QE is always close to the same value. This result is confirmed also by the study in [170, 171].

Another parameter used to estimate the minimum number of nodes is the *mean local error*. It is the mean distance between the last 500 BMUs and the corresponding input values. Differently



(a) (b)

Figure 5.24.: PC in input to each test (a). QE in relation to N (b).

from the QE, now the output value is more variable and related to the last updated nodes. Finally, a third parameter is the *mean square error*, it is similar to QE, but it uses the square distance.

$\varepsilon_w - \varepsilon_n$

These parameters are used to weight the displacement of the BMU and its neighbours to the input, respectively (p.6 in Pseudocode 4.1). It means that their values must be in the range $[0-1]$ otherwise, if they are greater than 1 the node will go beyond ξ and for negative values the element will move in the opposite direction.

Values close to 1 could sometimes lead to an unstable network, because the system will be too sensible to a single input. On the other hand, low values are typical for a model that is not able to quickly adapt itself to the manifold. As stated in [153], normal values for ε_w are in the range $[0.3 - 0.05]$, whilst for ε_n, a ten time lower value is recommended. For this application has been selected ε_w= 0.1 and ε_n= 0.01.

λ

The GNG adds a node to the networks every λ iterations, then the parameter is related to the available time for the GNG. When this is not a restriction, λ can be set equal to one or two orders of magnitude less the dimension of the PC. According to this rule, the nodes are gradually added to the network in a proper manner, based on the criterion that reduces the global error. On the other hand, when there are some constraints due to the computational time, the parameter can be

set very low. For our purposes indeed, to have a frame rate of 30/15 fps, an adequate value for λ is 10. This configuration allows to have a network that fits very fast the input, but a quantization error slightly bigger than the normal situation.

Another consequence related to a low value of λ, that will be mentioned later, is the possibility that some nodes become inactive.

$\alpha - \beta$

They are the weight factors used to reduce the local error of a node. β is used at the end of each iteration for all the node errors and α for the two closest elements to a new node. They are respectively set to 0.5 and 0.0005 following the standard configuration published in [153, 155].

a_{max}

It is the maximum age allowed to an edge, a connection is deleted when its age exceeds a_{max}. At each iteration of the GNG, the age is incremented only for edges that start from the BMU. Therefore, for the same value of a_{max}, the probability to delete a connection is lower for a network that has more nodes than another. Therefore the parameter is strongly related to N and also to the total number of iterations per PC.

In this application, considering the previous network parameters, in most of the cases $a_{max} = 15$ is adequate to preserve the topology. Higher values (e.g. $a_{max} = 30$) can lead to connections that continue to be active also when their nodes are far away each other.

Stop Condition

The stop condition (p.11 in Pseudocode 4.1) can be reached if one of the following states is true:

- a criterion of error minimization is verified;
- the maximum number of nodes is reached;
- the processing time for single PC ends;
- all the points in PC are processed by the GNG.

The first condition requires a considerable time to be performed because, for example using the QE, for each iteration must be calculated Eq. (5.33). The second criterion instead is justifiable only in the first frame, when the system starts with a graph of two nodes. Indeed, in next PCs, the network is not restarted, but directly used as input to the GNG, as implemented in similar applications [155, 171]. Therefore, it will have always the maximum number of elements at the beginning of the interactions.

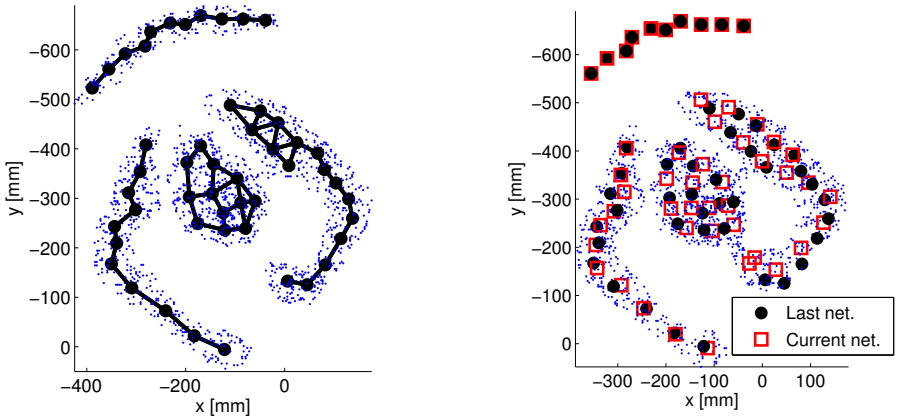
The processing time is another information used in literature as stop condition [152]. It is useful if project constraints limit the time interval to process the data or when the size of the manifold is variable or not known before starting the algorithm.

In view of these considerations, the fourth condition is used as stop criterion to GNG. In this way, processing all the elements in the manifold, the algorithm will be also able to take into account small differences between consecutive PCs.

Inactive Nodes

A node is classified as inactive if it is "too far away" from the PC and the probability to be selected as BMU is equal to zero. As introduced in the discussion of parameter λ, the original GNG algorithm does not solve the inactive nodes issue. This problem could be due to a wrong selection of the network parameters ($\varepsilon_w, \varepsilon_n, a_{max}$ are too small) or related to a fast change in the manifold distribution, for example when some outliers appear in input.

Figure 5.25 shows an example where inactive nodes are present. In particular, in Figure 5.25a is visible the PC in top view superimposed to the network. In Figure 5.25b, the chair PC is no longer present. This effect is due to the sensor configuration and to the foreground technique used to select the person's blob from the depth frame. The chair's nodes continue to be present, but they will not be selected as BMU because too far away from the PC (head, arms).



(a) (b)

Figure 5.25.: Example of inactive nodes. Network adapted to the last PC (a), current PC where the chair's nodes become inactive (b). The black nodes model the last PC whereas the red squares represent the current network.

This situation is a waste of resources because the inactive nodes are used to model an old element, while they could be exploited to improve the representation of other active parts. Furthermore, the quantization error is negatively influenced by this condition because there are less nodes in the current PC.

Fritzke has proposed a possible solution to this problem in [172]. The idea is to use another parameter for each node in addition to the accumulated error, to find the elements of the network

that contribute less to the reduction of the global error. These nodes are removed from their positions and added to the group of free nodes used in (p.9) in Pseudocode 4.1. The parameter exploited to find these nodes is called *Utility*.

$$U_{s1} = U_{s1} + \|w_{s2} - \xi\|^2 - \|w_{s1} - \xi\|^2 \tag{5.34}$$

Eq. (5.34) is implemented for each iteration, after the update of the accumulated error for the BMU (w_{s1}). The equation allows finding the nodes that are close each other or the isolated ones, i.e. elements that have a low probability to be selected as BMU.

When a new node is added, the *Utility* is updated as follows:

$$U_r = \frac{U_q + U_f}{2} \tag{5.35}$$

While the process to reduce all *Utility* values is identical to the operation (p.10) in Pseudocode 4.1.

$$\frac{Err_a}{U_r} > k, \quad Err_a > Err_j \quad j \in N \tag{5.36}$$

Finally, the node r, with the lowest value of *Utility*, can be deleted if it verifies the condition in Eq. (5.36). a is the index of the network element with the highest local error and k is an empirical constant. Low values of k are suitable to quickly delete the inactive nodes, but can affect also the useful ones. At the same time, high values of k can reduce the reactivity of the algorithm to find and delete inactive nodes. Unfortunately, there is not a general rule to properly set the k value. When the parameter is not correctly identified, a direct consequence can be an unstable network.

In this application, the problem has been solved exploiting an alternative approach based on a common property of the inactive nodes. An isolated node has, by definition, no chance to be a BMU. Consequently, its position will not be updated during all the time that the algorithm needs to process a single PC. The chair's nodes in Figure 5.25b confirm this, their positions are the same in the previous and current frame. As indicated in Eq. (5.37), when the analysis of a single PC is completed, the last network is compared with the previous one.

$$w_i(t_n) - w_i(t_{n-1}) = 0, \quad i = 1, 2, ..., N \tag{5.37}$$

If Eq. (5.37) is verified, the node is classified as isolated element.

Topology preservation

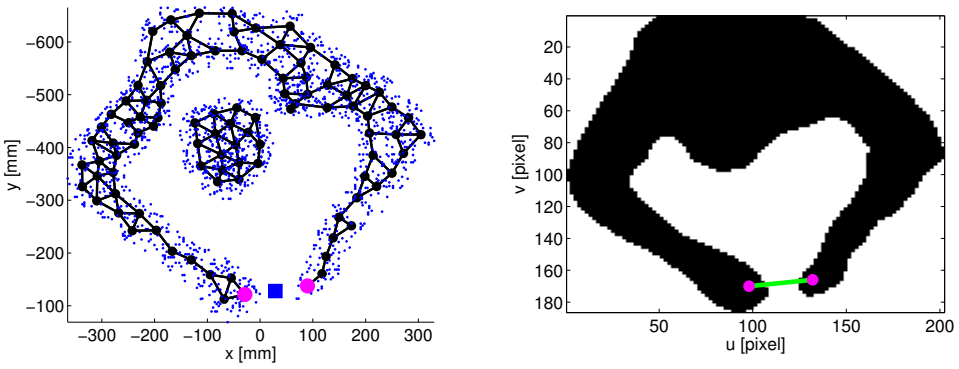
Another difference with the original GNG [150] is the analysis of the topology preservation of the network.

The standard algorithm has been modified to have a network as much as possible similar to the manifold. In Figure 5.26a is visible in top view the GNG output during the training session to the PC. The blue square represents the specific input (ξ), while magenta dots are s_1 and s_2,

respectively placed in the left and right hand. (p.7) in Pseudocode 4.1 forces the creation of a new connection between the nodes. As a consequence, the topology will not be preserved because there is an empty space between the hands.

As stated by Flórez-Revuelta et al. in [157], the information of the manifold has to be exploited to find if the topology preservation is verified. In this case, three alternatives are possible:

- use the definition of *Topographic product* and *Topographic function* introduced in Section 4.3.2;
- check in the PC if the density of the points between s_1 and s_2 is enough to allow a new connection;
- use the depth frame to assess if the connection is inside the blob (Figure 5.26b).



(a) (b)

Figure 5.26.: Trained network to the PC (a), corresponding blob in the depth frame (b).

Comparing the three solutions, the third one has a less computational cost than the other two. Indeed, for the first solution, the distance relations between all the reference vectors in the input space and all the nodes in the output space must be analysed. Whereas in the second case, each element in the PC must be evaluated to identify all points in the region between s_1 and s_2.

In the third case, it is sufficient to control that the green line in Figure 5.26b is inside the blob. As visible in the figure, this is not true and the connection is not established.

Using the previous discussion to tune the algorithm parameters, now the GNG algorithm can be set to model the PC calculated in Section 5.3.1. In Figure 5.27a is visible a generic PC provided in input to the GNG. The PC must be pre-processed (Figure 5.27b) before using the algorithm deleting the head points by a specific filtering operation (Eq. (5.38)). This simplifies the post-processing code, that will be described later, used to identify the hands joints starting from the network in output from the GNG.

$$\|PC_{hd}(t) - J_{hd}(t-1)\| < th_{hd} \tag{5.38}$$

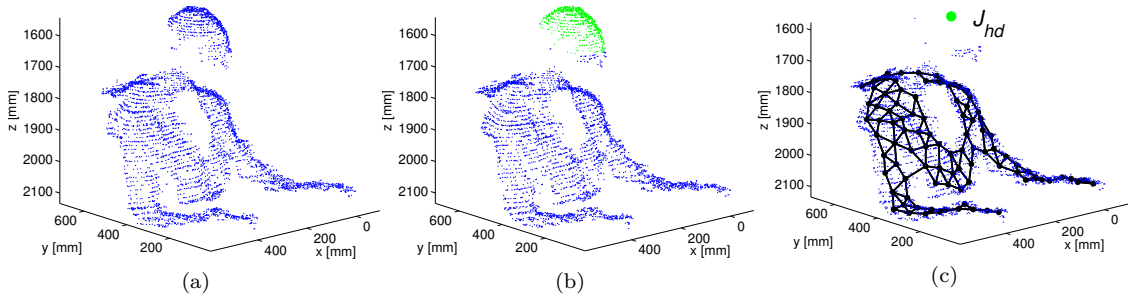


Figure 5.27.: Person's PC (a). Head's PC highlighted in green(b). Network provided by the GNG algorithm (c).

In Eq. (5.38), $PC_{hd}(t)$ represents the points of the head in the last PC, J_{hd} is the head joint calculated from the mean of all the elements in PC_{hd}. th_{hd} is the distance threshold used to find the head PC at t, based on the position of J_{hd} in the previous frame. In the first PC, J_{hd} is set as the highest point in the manifold, whilst for all the next time, J_{hd} is the centroid of $PC_{hd}(t)$.

The filtered PC will be the input to the GNG and in Figure 5.27c is visible the trained network in output.

In consideration of the aforesaid, Table 5.10 summarizes the parameters used by the GNG. The variable PC Step is the sampling factor applied to the PC. In particular, it means that considering 10 elements of the PC, only one of them is used as input to the GNG. In this way, the total processing time required by the algorithm is less than 40/50 ms, i.e. it is compatible with a frame rate of 15/20 fps.

PC Step	Net. size	ε_w	ε_n	λ	α	β	a_{max}
10	100	0.1	0.01	10	0.5	0.0005	15

Table 5.10.: Selected parameters for GNG algorithm.

5.3.3. Hand Joints Recognition

When the training phase is completed, each self-organizing algorithm previously tuned provides in output a trained network. The next step is to find the nodes that model the hands. Indeed, these nodes are essential to understand the movements performed by the person. For the SOM, in all the networks, the hand joints match with the nodes labelled as w_{16} and w_{17}. Similarly, for the SOM_Ex they are the elements at the end of the planes that model the hand.

On the contrary for the GNG, during the training phase of the model with different PCs, each node could significantly change its position and number of connections. For example, a node close to the hand position, in next networks could move near the chest area, or be deleted if the condition

(p.8) in Pseudocode 4.1 is verified. Therefore, there is not a predefined node index for the element that models a hand. To solve this problem, an appropriate post-processing algorithm is devised. It starts from the network in output from the GNG to recognize the hand joints among all the nodes in the network.

The network theory can be exploited to solve this task and some functions of graph analysis have been used to implement the solution described below. In particular, the network provided by the GNG can be represented as a graph [173] with the following features:

- *undirected*: the connections are bidirectional;
- *unweighted*: each connection has not an associate value;
- *cyclic*: in the network there are some closed loops formed by more than two nodes i.e. starting from an element, there is a path to return to the same node;
- *disconnected*: some nodes are isolated and can not be reached from a generic point. For example, the head's nodes in Figure 5.26a are not connected to the body's nodes;
- *no self-loops*: there are no single connections that start and end in the same node;
- *no multi-connections*: each nodes pair has a single connection.

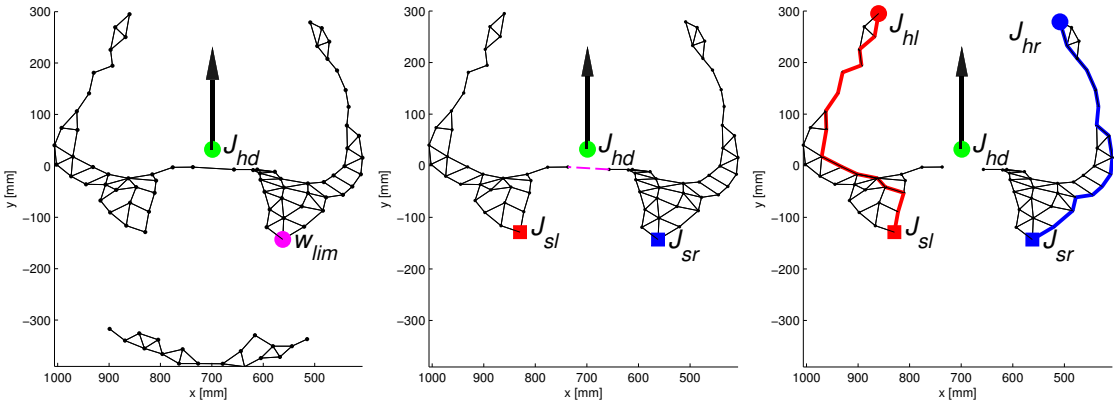
As visible, in Figure 5.28a, the network in output from the GNG has been rotated in the standard direction inverting Eq. (5.26). Consequently, the network direction (black arrow) is always parallel to the y-axis, independently of how the person is oriented with respect to the camera. The green point is the joint J_{hd} calculated by Eq. (5.38).

The Pseudocode 5.5 describes the operations to identify the hand joints. The first step is to find all the clusters in the network (p.1), i.e. the groups of nodes connected by edges. In the graph theory, an efficient algorithm that solves this task is presented in [174]. In particular in Figure 5.28a, the algorithm provides two groups, the first one models the person and the second one the chair. The cluster with the maximum number of nodes (C_{big}) is selected.

The node w_{lim}, highlighted in magenta in Figure 5.28a, is obtained from the line (p.3) as the lowest element of C_{big}. It will be used to separate the useful clusters (C_{pers}) from the other ones (outliers). The result is shown in Figure 5.28b where the chair's points are deleted. The network in Figure 5.25a can be used for a second example. In that case, there are 4 clusters, three of them model the person and one represents the chair PC. The network must be rotated in the standard direction and then, exploiting lines (p.4-8), the chair can be discarded. The code marks a cluster as useful if it has at least one node above w_{lim} with respect to the y direction. In this way, the arms and the head clusters in Figure 5.25a are correctly identified.

The shoulder joints are selected in lines (p.9-10), respectively as left and right node to J_{hd} with the lowest y coordinate from all elements in C_{pers}. These two joints are highlighted in Figure 5.28b with coloured squares.

Assuming that the hand joints are the furthest points from the corresponding shoulder joints (J_{sl}/J_{sr}), the well-known Dijkstra's algorithm [175] can be exploited to find these elements.



(a) (b) (c)

Figure 5.28.: Algorithm used to find the hand nodes inside the network. Identification of the point w_{lim} (a), the outlier nodes and the chest's connections are deleted (b), identification of the hand joints (c).

Algorithm 5.5 Algorithm used to find the shoulder and hand joints.

Input: (Net_{GNG}, J_{hd})
Output: (J_{hl}, J_{hr})

$[C_1, C_2, ..., C_N] = findClusters(Net_{GNG})$
$C_{big} : dim(C_{big}) > dim(C_i),\ \ i = 1, 2, ..., N_{clust}\ \ i \neq big$
$w_{lim} : w^y_{lim} < w^y_i,\ \ i = 1, 2, ..., dim(C_{big})\ \ i \neq lim$
for $e = 1 : N_{clust}$ **do**
  if $\exists w_i \in C_e : w^y_{lim} < w^y_i,\ \ i = 1, 2, ..., dim(C_e)$ **then**
    Add C_e to C_{pers}
  end if
end for
$J_{sl} : w^x_i < J^x_{hd}$ **and** $J^y_{sl} < w^y_i\ \ \forall i = 1, 2, ..., dim(C_{pers})$
$J_{sr} : w^x_i > J^x_{hd}$ **and** $J^y_{sr} < w^y_i\ \ \forall i = 1, 2, ..., dim(C_{pers})$
for $i = 1 : dim(C_{pers})$ **do**
  w_{nrb} are the neighbours of w_i
  if $w^y_i < J^y_{hd}$ **and** $(w^x_i > J^x_{hd}$ **and** $w^x_{nrb} < J^x_{hd})$ **or** $(w^x_i < J^x_{hd}$ **and** $w^x_{nrb} > J^x_{hd})$ **then**
    Delete the connection in C_{pers}
  end if
end for
$J_{hl} = Dijkstra(J_{sl}, C_{pers}),\ \ J_{hr} = Dijkstra(J_{sr}, C_{pers})$

Indeed, given the path with the most number of hops between nodes, the hand will be its last element.

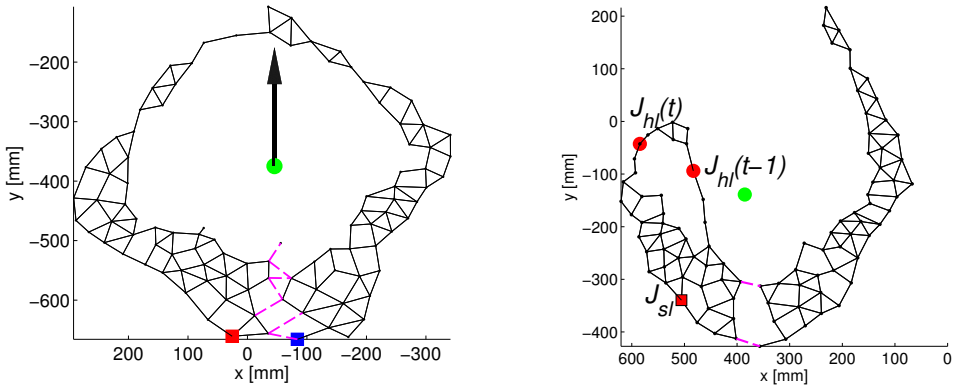
Before finding the hand nodes, as indicated in (p.11-16), it is necessary to delete the chest's connections that link the left part of the cluster to the other one. As visible in Figure 5.28b, there is only a single connection to discard, highlighted with a magenta dashed line. Without this alteration to the network, the Dijkstra's algorithm could give in output a wrong path. Figure 5.28c shows the result, where the hand joints are labelled with J_{hl}/J_{hr} and the paths provided by the Dijkstra's algorithm are highlighted.

In Figure 5.29a is visible a critical situation: the arms are joined together in a single chain and, as a consequence, the Dijkstra's algorithm fails to find the joints J_{hl} and J_{hr}. Anyway, the situation can be easily identified because, only in this case, after the code (p.11-16) is executed, there is still a direct path that joins J_{sl} to J_{sr}. When this condition is verified, the left hand joint is calculated using an alternative equation. It will be the node ($w_e(t)$) closest to the left hand joint of the previous graph ($J_{hl}(t-1)$):

$$\begin{aligned} J_{hl}(t) = w_e(t) : \|w_e(t) - J_{hl}(t-1)\| < \|w_i(t) - J_{hl}(t-1)\| \\ i = 1, 2, ..., dim(C_{pers}) \end{aligned} \tag{5.39}$$

The same formula can be used to find the right hand node, replacing J_{hl} with J_{hr}.

Figure 5.29b shows another case where the Dijkstra's algorithm cannot be directly used. The person's hand is very close to the face and sometimes the left arm creates a closed loop with the shoulder. In this condition, the farthest node to J_{sl} is not placed near the hand.



(a) (b)

Figure 5.29.: Critical situations. Fusion between hand joints (a), closed loop for left arm (b).

The Pseudocode 5.6 is used to find and solve the problem and it is run always after the Pseudocode 5.5. The algorithm accepts in input:

– the cluster C_{pers};

– the shoulder joint;

– the hand joint in the current and last network.

When the condition (p.1) is verified, it means that the left hand joint in the last network ($J_{hl}(t-1)$), is above the shoulder ($J^z_{sl}(t)$) and far away from the position found in the current graph ($J_{hl}(t)$). Then, the Dijkstra's algorithm is used to find a possible path between $J_{sl}(t)$ and $J_{hl}(t-1)$. If the path exists, and all its nodes are sufficiently far away from $J_{hl}(t)$, the closed loop is identified and it must be handled.

To this end, $J_{hl}(t)$ is set as the closest node to the sensor in $Path_{lft}$, because in this particular pose the hand is the highest part of the arm. The code for the right hand is the same, but J_{hr} and J_{sr} substitute J_{hl} and J_{sl}, respectively.

Algorithm 5.6 Code to handle the closed loop issue.

Input: $(C_{pers}, J_{sl}(t), J_{hl}(t), J_{hl}(t-1))$
Output: $J_{hl}(t)$

1: **if** $J^z_{hl}(t-1) < J^z_{sl}(t)$ **and** $\|J_{hl}(t-1) - J_{hl}(t)\| > th_1$ **then**
2: $Path_{lft} = Dijkstra(J_{sl}(t), J_{hl}(t-1), C_{pers}(t))$
3: **if** $\|w_i - J_{hl}(t)\| > th_2 \ \forall i \in Path_{lft}$ **then**
4: close loop found
5: J_{hl} is set as the closest node to the sensor in $Path_{lft}$
6: **end if**
7: **end if**

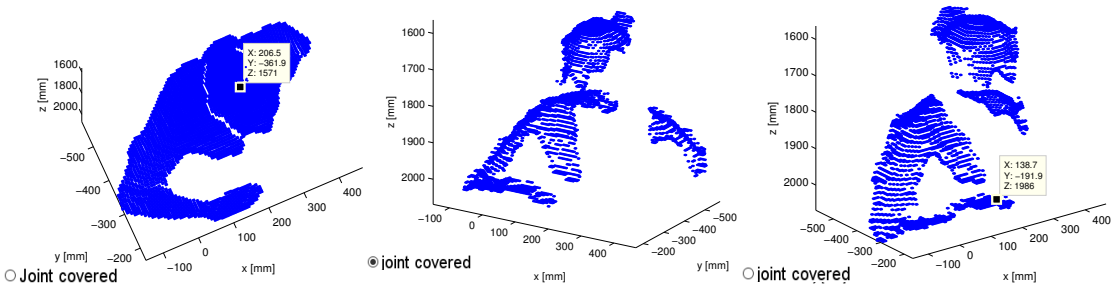
5.3.4. Tracking Performance

The three self-organizing algorithms have been used to track the movements performed by all the volunteers in two datasets, for a total of 55 people. In particular, for each PC are available the positions of the three most important joints: head (J_{hd}), left hand (J_{hl}) and right hand (J_{hr}). These values are useful to identify most of the actions performed by a person during a meal. For example, the distances $\|J_{hd} - J_{hl}\|$ and $\|J_{hd} - J_{hr}\|$ are used to find when a hand is very close to the face. Alternatively, when some objects are on the table, it is possible to monitor the interactions between them and the person.

In order to find the algorithm that has the best accuracy, it is necessary to compare the joint positions with a ground truth. Unfortunately, during the recording of the dataset *Dts_intake_*1, the use of physical markers has not been planned. Therefore, their positions ($J_{hd,GT}, J_{hl,GT}, J_{hr,GT}$) have been manually assigned evaluating the PC using a custom Matlab tool and taking into account that when an area of interest is not visible, for example due to an occlusion, the joint is not labelled.

Figure 5.30 shows the plots used to label the positions of the useful joints. For the left hand (Figure 5.30b), this operation is not possible since the arm is partially covered by the shoulder. In

this case, the joint is marked as "not identified". In total, 9.5 K frames have been analysed and 9.4 K - 8.8 K - 6.3 K ground truth positions are respectively assigned as $J_{hd,GT}$, $J_{hl,GT}$ and $J_{hr,GT}$.



(a) (b) (c)

Figure 5.30.: Matlab tool used to find the ground truth. Plots for head (a), left hand (b) and right hand (c).

On the other hand, for the dataset *Dts_intake_2*, three IR markers have been exploited to find automatically the ground truth. In particular, in the IR frames, the colour of these objects is brighter than the other regions of the image, therefore an algorithm can easily identify their coordinates. The corresponding 3D positions are calculated using the intrinsic depth camera parameters of the IR sensor. In this case, for each joint, 60 K coordinates are found.

The performances of each algorithm are evaluated using the Euclidean distance in order to find the gap between the joints position and the corresponding ground truth. For the dataset *Dts_intake_1*, all the 48 tests are considered as a single sequence, taking into account the indices where the ground truth is available.

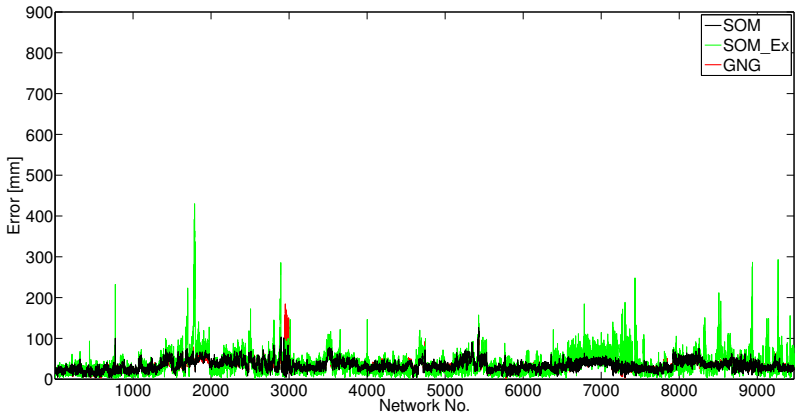
$$err_{pos,alg} = \|J_{pos,alg} - J_{pos,GT}\| \qquad \begin{array}{l} pos = \{hd, hl, hr\} \\ alg = \{GNG, SOM, SOM\_Ex\} \end{array} \tag{5.40}$$

In Eq. (5.40), err is the distance between the joint coordinates estimated by the algorithm ($J_{pos,alg}$) and the real position ($J_{pos,GT}$). For each error trend has been calculated the mean (μ) and the standard deviation (σ).

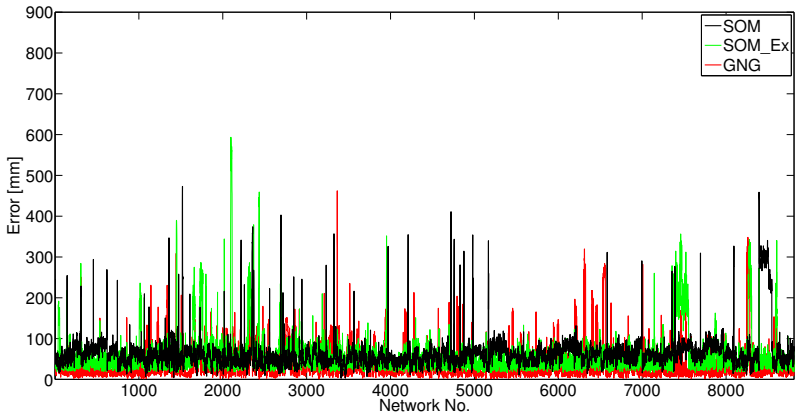
Figure 5.31 shows the complete error trends for the three joints. The x-axes have different ranges because the number of available ground truth positions is not the same in the three situations. On the other hand, the y-axis have the same maximum in order to compare the error trends.

Concerning the head, there is not a noticeable difference between the SOM and GNG curves. Indeed, they are overlapped and most of the time below the 100 mm. Whereas, the trend for the SOM_Ex is affected by a more variability, with peaks higher than 200 mm.

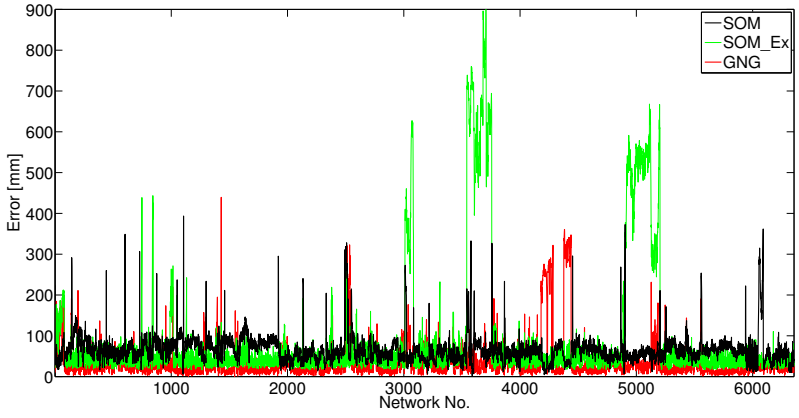
The left hand is characterized by a marked variability than the head, for all the three joints.



(a)



(b)



(c)

Figure 5.31.: Error trends for the head (a), left (b) and right (c) joints for *Dts_intake_1*.

This is caused by the movements of the hands during intake activities (eating, drinking, pouring water and moving objects on the table). Whereas, the head usually has a less displacement, limited to move forward for eating or backward for drinking.

Figure 5.31b shows that, most of the time, the GNG has the best performances. It is most of the time below the error trend of the SOM_Ex, whilst the SOM has the higher error. The peaks for the SOM and SOM_Ex are due to a wrong fitting of the network to the PC that leads the hand joints to be placed away from the correct positions. The GNG peaks are caused by the challenging configuration of the trained network that leads the Dijkstra's algorithm to find a wrong node for the hand position. However, these peaks refers to tracking problems that affect single networks and most of the time the algorithms give in output the correct positions.

The same consideration is not true for $err_{hl,SOM}$ because, around the sample 8.2 K as visible in Figure 5.31b, there is a high error that continues for more networks. In that situation, while the person is drinking using the left arm, the nodes chain that models the limb is attracted by the PC of the other arm and consequently, J_{hl} is placed in the right arm.

A similar problem affects err_{hr,SOM_Ex}. In particular, it is shown in Figure 5.31c around indexes 3 K, 3.7 K and 5 K.

The error ranking of J_{hl} is similar for J_{hr}, where the GNG has a lower error trend than the other two algorithms.

In Table 5.11 are visible the statistic parameters for the three joints. As confirmed by the previous consideration, in all the situations the GNG algorithm has a lower error than the SOM-SOM_Ex ones.

Joints\Alg	**GNG**		**SOM**		**SOM_Ex**	
	μ [mm]	σ [mm]	μ [mm]	σ [mm]	μ [mm]	σ [mm]
Head	**33.2**	**13**	33.3	12.4	38.4	28
Left hand	**37**	**38.1**	64.8	43.7	42.3	49
Right hand	**36.9**	**49.2**	66.35	35.13	78.2	148.8

Table 5.11.: Mean and standard deviation for distance errors using *Dts_intake_1*.

In the light of the above error trends, the low variability of the head joints is confirmed by the lower standard deviation for $J_{hd,alg}$ than the other two joints.

The right hand joint for the SOM_Ex has a mean error of 78 mm and a significant variability (148 mm) due to a high error trend in the three intervals where err_{hr,SOM_Ex} is above 400 mm (Figure 5.31c).

The statistical proof that there is a considerable difference in the error values is assessed by the ANOVA test (*Analysis of Variance*). In particular, the test has been used for each joint, comparing the following pairs $(err_{pos,GNG}, err_{pos,SOM})$, $(err_{pos,GNG}, err_{pos,SOM\_Ex})$ and $(err_{pos,SOM}, err_{pos,SOM\_Ex})$. Only for $(err_{hd,GNG}, err_{hd,SOM})$ is not possible to assert a significant difference. Indeed, as visible in the first row of Table 5.12, the F value is lower than F

crit. For all the other cases, the ANOVA test gives positive results.

Output Measure	Source of variation	Sum of Sq.	DOF	Mean Sq.	F	F crit	p-value
$(err_{hd,GNG}, err_{hd,SOM})$	Between groups	23.035	1	23.03	0.14	3.84	0.70
	Within groups	3064k	18k	161.7			
$(err_{hl,GNG}, err_{hl,SOM})$	Between groups	3415k	1	3415k	2026	3.84	0
	Within groups	29691k	17k	1685			
$(err_{hr,GNG}, err_{hr,SOM})$	Between groups	2749k	1	2749k	1503	3.84	0
	Within groups	23248k	12k	1829			

Table 5.12.: ANOVA test results for $(err_{pos,GNG}, err_{pos,SOM})$ using *Dts_intake_1*.

The results for the dataset *Dts_intake_2* are summarized in Table 5.13. The good performances of the GNG are confirmed also in this case. It outperforms the other two algorithms two times out of three. Only for the head joint the SOM has a lower mean error, but the difference is negligible.

The ANOVA test is repeated also in this case and the F value is always higher than the F crit in all the possible combinations, therefore we can state that the statistical results are significant.

Joints\Alg	**GNG**		**SOM**		**SOM_Ex**	
	μ [mm]	σ [mm]	μ [mm]	σ [mm]	μ [mm]	σ [mm]
Head	43.8	17	**42**	**19**	61	86.6
Left hand	**51**	**44**	82	47	67	84.9
Right hand	**47.7**	**33**	89.5	37.9	57.5	42.5

Table 5.13.: Mean and standard deviation for distance errors using *Dts_intake_2*.

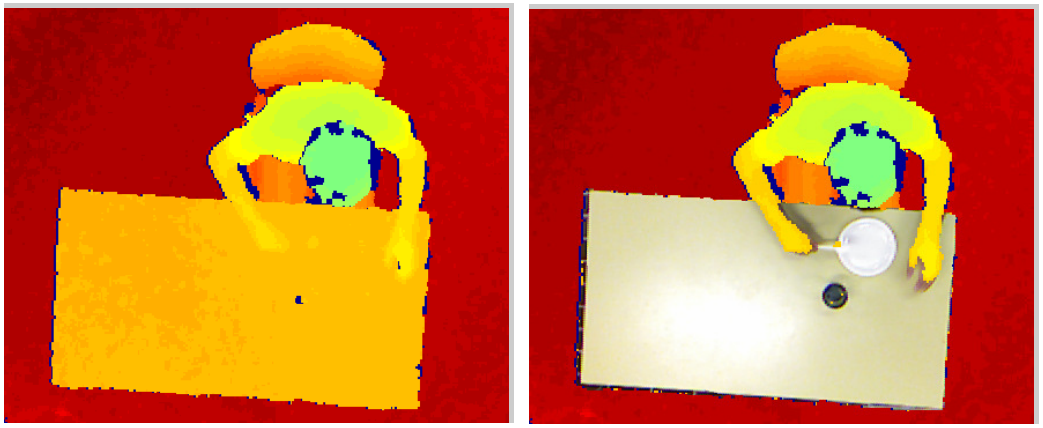
5.3.5. Processing of the RGB Stream

Another useful data exploited by the algorithm is the RGB stream. Indeed, the colour information provided by Kinect is used to find useful objects on the table and takes part in the final decision to infer intake actions.

The calibration algorithm described in Section 3.2 is used to map the depth frame in the RGB one. In particular, for each depth pixel, the equivalent one in the colour frame is identified. After this operation, the colour stream is ready to compensate some limitations of the range sensor. In

fact, the configuration of the sensor does not allow discerning the presence of objects like plates or glasses. Figure 5.32a highlights the problem: using the depth information only, it is not possible to identify some elements near the subject. On the other hand, the objects are easily visible in the mapped RGB table.

In order to achieve the result shown in Figure 5.32b, the depth pixels of the table, detected in Section 5.3.1 to find the person's direction, are mapped in the RGB frame. The person's PC, superimposed to the table, is then discarded from the RGB frame. Now the system is ready to scan the RGB table pixels ($tableRGB$) to find glasses.



(a) (b)

Figure 5.32.: Raw depth frame, where dishes on the table are not detectable (a). Depth frames with mapped RGB table, where a glass and a plate are visible (b).

The circular properties of these objects are exploited by the Hough function (Section A.3) and their central coordinates are provided in output. An efficient implementation of the Hough function integrated in the OpenCV library is presented in [93, pp. 158].

The grabber in [176] allows to import this library in Matlab, so that the parameters used by the Hough function can be tuned. The function uses in input:

- area of interest ($tableRGB$);
- Canny threshold [pixel];
- minimum/maximum radius of the circle [pixel];
- accumulator plane threshold [pixel].

The glass radius is set to 35 ± 10 mm and it represents an adequate range in a real case. This interval must be converted to pixel data considering that it changes with respect to the distance

between the object and the sensor (dpt). By inverting Eq. (5.4) and using a depth value around the table, the pixel range (wd_p) is related to the real one (wd_r):

$$\frac{wd_p}{320} = \frac{wd_r}{2bs} \rightarrow wd_p = \frac{320 \cdot wd_r}{2bs} \qquad bs = dpt \cdot \tan(28.5^\circ) \tag{5.41}$$

For example, in Figure 5.32b, the table is 2082 mm far from the device and then the calculated glass radius is 5 ± 1 pixels. The Canny threshold is set to a percentage of the average grayscale histogram of $tableRGB$. Finally, the accumulator plane threshold is the circumference of the circle to search in the Hough domain, expressed in pixel. Considering the previous example, featuring a glass radius of 5 pixels, the accumulator plane threshold is $2\pi(5 \pm 1)$ pixel. As result of the RGB processing, the algorithm is able to identify the coordinates of circular objects in the RGB table area. It is worth noting that the system automatically sets the above variables using only the depth value of the table, so that the Hough parameters can be easily adapted to different height set-ups.

If the object to find is a plate, the same algorithm can be exploited, but the radius must be changed to 150 ± 50 mm to take into account the different object's size compared to the glass.

5.3.6. Drinking Recognition

The tracking solution to find the hands coordinates in the 3D space along with the information of useful objects on the table are combined together to infer the drinking activities.

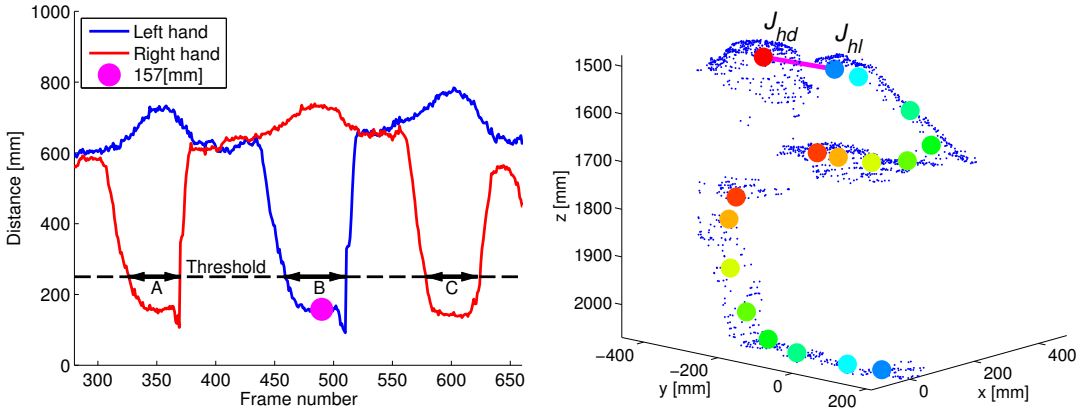
The dataset used is the Dts_intake_1 and the joints coordinates (J_{hd}, J_{hl}, J_{hr}) in output from the SOM algorithm are evaluated. For each frame, the distances between the hands and the head are monitored:

$$Dist_{left} = \|J_{hd} - J_{hl}\| \qquad Dist_{rgt} = \|J_{hd} - J_{hr}\| \tag{5.42}$$

Figure 5.33a shows $Dist_{lft}$ and $Dist_{rgt}$ trends for a test execution. The magenta point highlights the situation when the hand is very close to the head. In particular, the distance $Dist_{lft}$ at 490^{th} frame is 157 mm. The corresponding PC related to this situation is visible in Figure 5.33b, where the magenta segment that connects J_{hd} and J_{hl} is $Dist_{lft}$ at 490^{th} frame. A threshold set to 250 mm allows to select useful movements.

The colour information is then exploited to recognize drinking actions. The solution previously described provides, when present, the coordinates of any glass near to the person. In view of these facts the algorithm, at the first useful frame in input to the tracking solution, searches the glass on the table and stores the average grayscale histogram of the object. In next frames, when the distance $Dist$ decreases below the threshold, the algorithm tries to find the same object.

In details, in Figure 5.33a the system processes the RGB frame in the intervals A-B-C. Figure 5.34 points out four portions of the RGB frames 150, 350, 490, 600, where only the table is visible. At the 150^{th} frame, the algorithm stores the colour information of the glass. Subsequently, at the 350^{th} frame, as visible in Figure 5.33a, the distance $Dist_{rgt}$ decreases very fast. To confirm the drinking action, the algorithm processes the 350^{th} RGB frame. Indeed, in Figure 5.34b the glass

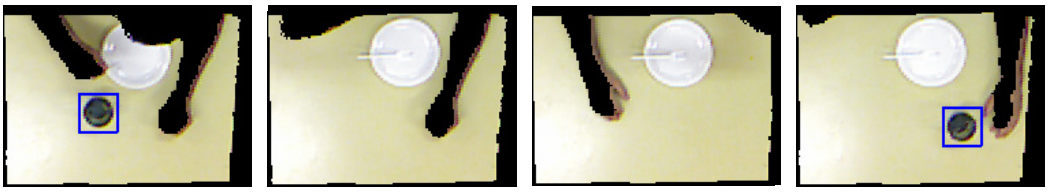


(a) (b)

Figure 5.33.: Left and right hand-head joint distances during a test execution (a). A distance threshold is exploited to recognize the proximity of head and hand joints. At 490^{th} frame, a drink intake action is reported. PC of 490^{th} frame where the left hand is close to head (b).

is not present on the table and then the action is correctly identified. At 490^{th} frame the same situation is repeated, but now the subject uses the left hand. Finally, at frame 600^{th}, $Dist_{rgt}$ value is similar to previous ones, but now, as visible in Figure 5.34d, the RGB control recognizes the glass. The algorithm classifies this action as "not drinking", because the subject takes the hand to the face without the glass. 38 out of 48 situations are used as input to the data fusion algorithm, for a total number of 35 different people. For the other 10 tests, the RGB streams are not available.

Figure 5.35a shows, for tests from 1 to 20, the system output during the total period of each single analysis. When there are no coloured bars, it means that $Dist_{rgt}$ and $Dist_{lft}$ do not satisfy the distance check. In frames labelled as "hand gesture" the hand is close to the head, but the glass



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

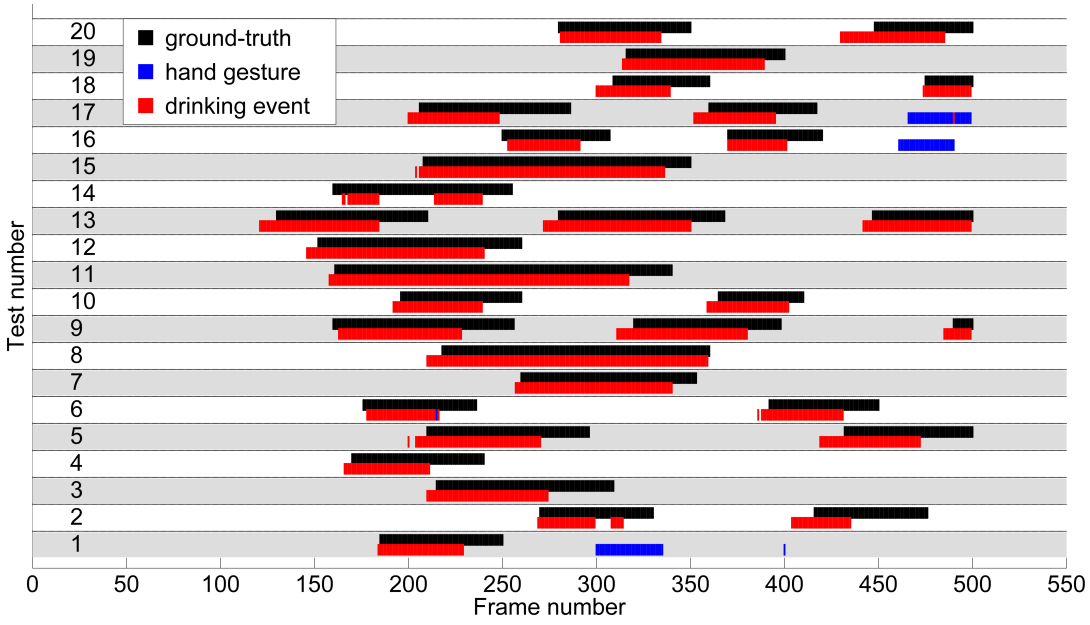
Figure 5.34.: Sequence of RGB frames, processed to recognize the presence of glasses. Glass detected in the initial RGB frame (a). The glass is then searched in frames where $Dist_{lft}$ or $Dist_{rgt}$ are below the threshold: 350^{th} frame (b) 490^{th} frame (c) 600^{th} frame (d).

is on the table. Otherwise, when the situation is similar to the one described in Figure 5.34b-c the bars are labelled as "drinking event".

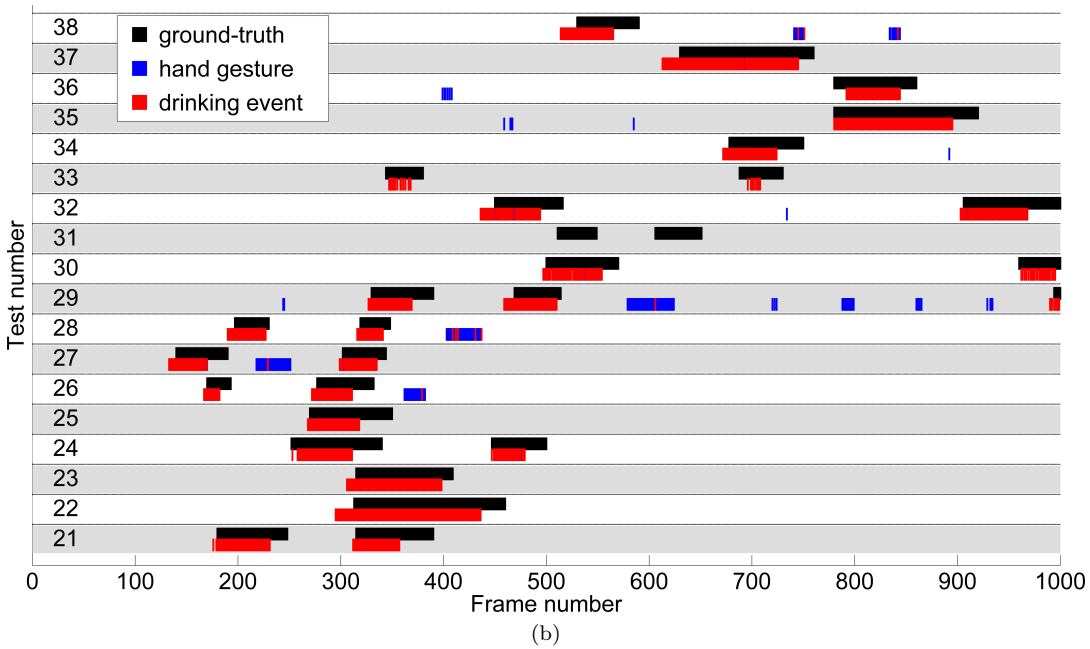
To confirm the result accuracy, each frame has been manually labelled as "ground-truth" or not. In details, the sequences of important frames start when the hand is very close to the face until the glass is released. In Figure 5.35a all the actions are correctly identified. Only tests 2 and 14 denote a different behaviour from the "ground truth". In those cases, some frames are not recognized as "drinking event" because the distance condition is not verified. For instance, considering the 14^{th} test, there is only a drinking action that covers the frame interval {160 - 255}. The system recognizes two different drinking actions because the distance conditions in Eq. (5.42) are not verified in the frame interval {184 - 214}.

The second part of the results is visible in Figure 5.35b. From the 29^{th} test, the total number of processed frames is increased to 1000. False positives occur in 26^{th} - 27^{th} - 28^{th} - 29^{th} tests because, although the distance control is verified, the algorithm has not been able to recognize the glass on the table. However, inside "hand gesture" bars, this situation is restricted to few frames. In particular 4% 2% 13% 2% respectively for the four tests.

The 31^{st} test is the only situation where the system is not able to identify the intake action. It is related to $Dist_{rgt}$ that is greater than the distance threshold even when the subject takes the hand to the head. Finally, considering the total number of tests, the algorithm correctly classifies actions with a percentage of 98.3%.



(a)



(b)

Figure 5.35.: Result scheme for tests from 1^{th} to 20^{th} (a). Result scheme for tests from 21^{th} to 38^{th} (b). The overlaps between "ground-truth" and "drinking event" labels denote the algorithm efficiency.

Chapter 6.

Conclusions

In this thesis, different AAL applications have been studied and implemented. They use information provided by wearable sensors and RGB-Depth cameras. In Chapter 2 are given some fundamental principles, in particular are described the most important technologies for the distance measurement and the features of Kinect V1 and V2. A similar description is repeated for the other sensor family, where the most important characteristics of the SHIMMER device are highlighted. At the end of the chapter, two recording tools have been presented to save the heterogeneous stream provided by the sensors. Thanks to these tools, five datasets are recorded and they are used to devise three AAL solutions.

Chapter 3 regards the synchronization issues between the data recorded by wearable and camera sensors. On the one hand, the BT wireless protocol introduces a variable delay in the packets sent by the SHIMMER device to the receiving system. A linear regression algorithm is used to solve this problem. It takes into account both the arrival time and the sampling rate to separate the received packets of the same time interval. On the other hand, the exposure and transmission time of the Kinect sensors are calculated for different streams (IR, depth and RGB) in order to know the time when an image is framed by the sensors and received by the program, respectively. Once these operations are implemented, each acceleration sample can be linked with an image by using the synchronized reference time.

When the RGB and depth stream are used in the same application, a synchronization between the frames is required. However, in this situation the problem affects the spatial correspondence between the pixel. The raw RGB and depth frames are modified using mapping functions that consider the locations of the sensors in the space. The result is two frames perfectly overlapped. The performances are compared with other four algorithms and the results show that the proposed mapping solution outperforms the other with a maximum error under 2 pixels.

The synchronization data are used to feed the algorithms designed for AAL. The first proposed application is a fall detection system presented in Chapter 5. It monitors the person's blob inside the depth image framed by Kinect V1 placed on the ceiling. The system triggers an alarm when the range between the person's centroid and the camera is similar to the floor distance. The application is able to track more than one person at time and to handle fusions between blobs.

A similar solution, that uses skeleton data from Kinect V2 and accelerations from two SHIMMER sensors, is proposed to classify different falls (forward, backward, side and end up sitting) from

ADL (sitting, crouching, walking and laying). In particular, three different algorithms have been compared and the best results are produced by the solution that monitors the skeleton joints displacement, the acceleration magnitude peaks and the distance between skeleton and floor. The algorithm is tested with the dataset $Dts\_ADLFall$ and the performances in terms of accuracy, specificity and sensitivity are 99% 100% 99%, respectively.

In order to assess the fall risk in the elderly, a person has been monitored during a TUG test through a SHIMMER device placed on the chest and Kinect V2. The data have been used to infer important parameters related to the fall risk, such as cadence time, stride length, maximum torso's inclination and time intervals of TUG phases. They are calculated for each one of the 20 volunteers in the dataset Dts_TUG to find statistical results.

The last application monitors the intake activities using RGB and depth streams from Kinect V1 in top-down view. The RGB is exploited to find specific objects on the table, while three self-organizing algorithms (SOM, SOM_Ex and GNG), introduced in Chapter 4, fit the same number of models to the person's PC. The coordinates of the most important joints (hands and head) are monitored to find when the person brings the hands to the face. The system combines the results of both the streams to infer important activities such as drinking. The performances of the fitting solutions, tested with datasets Dts_intake_1 and Dts_intake_2, are compared with a ground truth. The GNG algorithm outperforms the other two with a mean error below 52 mm for all the joints J_{hd}, J_{hl} and J_{hr}.

The finale system is tested with 38 examples in Dts_intake_1 and the drinking activities are correctly identified with a percentage of 98.3%.

6.1. Contributions

The contributions of the present research work are:

- a published tool (*Complete Viewer V2*) that stores the most important streams of Kinect V2 (IR, Depth, RGB, mapping matrix and skeleton) and related timestamps. It has been exploited to record 2 out of 5 datasets published online and used in this work. Other research groups can use the tool to record and publish their own datasets in order to test the algorithms performances with the same input data;

- a synchronization solution that correlates the streams from Kinect V1 (or V2) and the acceleration samples from wearable devices with the reference time of the Pc that handles the communications between the sensors;

- data fusion applications (fall detection and TUG test) that use the data from Kinect V2 and SHIMMER at the same time;

- study and implementation of three self-organizing algorithms for intake monitoring that use in input the PC in top view.

6.2. Future Research Directions

The results achieved in the different applications are promising, but a lot of work have to be carried out. For each application will be discussed the possible further research directions. In particular, in the fall detection system that uses the depth stream, the foreground elements are identified with a comparison between the depth map and a static reference frame. A possible improvement could be the update of this image to take into account dynamic changes in the background, such as furniture that is moved in the monitored area. For the person identification, it has been used a finite-state machine that controls 3D anthropometric characteristics of the person's silhouette. Another solution for the person recognition can exploit supervised machine learning approaches. For example, an Adaboost training process could be used to find the most representative Haar features in the depth map of the person and provides a classifier that discerns useful blobs from the other. Finally, the system to detect a fall, in addition to the height of the person's centroid from the floor, can monitor other features such as the centroid velocity and the body movements after the fall.

In the fall detection application, that uses the data from camera and wearable devices, the results can be compared with the performance of a machine learning algorithm. In particular, a linear SVM could classify fall and ADL using both the acceleration and the skeleton samples. Due to the limited operative range of the skeleton algorithm and its accuracy reduction when the person is lying on the floor, an alternative solution is to use directly the depth information to infer the conditions of the person. In this way, the depth camera can be placed in a different configuration from the standard one in order to cover more area and be less invasive for the user. In addition to the accelerometer, other sensors could be integrated in the wearable device such as gyroscope, magnetometer and barometer. The data fusion algorithm will take advantage of these measurements to improve the accuracy of the fall detection system.

The same considerations are valid also for the TUG test application. In particular, a wearable device with the sensors previously described can improve the recognition of both the first and the second turn phases. The next step is to use the parameters in output from the algorithm to assess automatically the fall risk factor in the person that has performed the test. To this aim, a new dataset is necessary, where healthy and high fall risk people must be recruited. A linear SVM, or other similar machine learning algorithms, will be trained with the results of the TUG algorithm using both the two groups. The complete system will be able to assess the fall risk to support the final decision of the medical staff.

In the intake monitoring application, as discussed in the section reserved to the performances of the self-organizing algorithms, not always the networks is correctly fitted with the PC. Therefore, in these cases the movement tracking is not possible. An alternative research direction could be the Random Forest algorithm used by the skeleton tracking in the official SDK.

In addition, now the system considers the distances $Dist_{lft}$ and $Dist_{rgt}$ to recognize when a hand is close to the face. Another strategy is to train a linear SVM to classify the networks in the same way. The advantage is that the limitation of a threshold base rule is overcome. Regarding the RGB stream, now the system recognizes objects like plates and glasses. In the future, more

sophisticated algorithms could be exploited to find different items such as fork, spoon and knife or directly the food inside the dish. However, considering the system set-up and the resolution of the RGB camera, the recognition of these elements is challenging. The 1920×1080 RGB resolution of Kinect V2 could overcome this problem, therefore it will be used in next datasets for the food intake monitoring.

Finally, all the proposed solutions should be tested, when possible, with similar datasets published on-line in order to evaluate algorithm performances with other input data. Generally, the proposed datasets are recorded in a laboratory that simulates real situations. Therefore, in the future, the applications must be tested in a real-world environment where volunteers are not young healthy people, but the elderly who will use this technology. In this way, the new set-ups will provide more realistic information about the system performances. This approach is already followed by some research institutes such as Eldertech-TigerPlace (Missouri University), Aware Home (Georgia Tech) and House_n (MIT) to mention a few of them.

6.3. Looking Forward: Trends and Technologies

In this work have been shown both the potentialities of wearable and vision technologies to tackle open problems in AAL. Regarding the wearable devices, they will be even more used in the future thanks to the boost carried out by the *internet of things* paradigm. The improvements in the integration process also lead the possibility to integrate more heterogeneous sensors in the same hardware, with the direct consequence that further information can be exploited by the embedded algorithms.

However, the most important open issue in this moment is the autonomy of these systems. Indeed, in a typical AAL application that monitors the health status of the elderly, the device must be worn as much time as possible in order to provide reliable information. Its autonomy is strongly related to the type of sensors integrated and the wireless protocols used to communicate the results on the processed data. These messages can be gathered by a wireless hub inside the smart environment or directly sent to informal caregivers using a mobile network. Today, the result of this situation is a trade-off between the processing capabilities and autonomy. In the future, if the charging issue will be overcome, for instance by using more sophisticated wireless charging solutions, these devices will gain more interest for the final users.

On the other hand, in the last years the depth sensor trend has moved in the reduction of the camera size. PrimeSense Capri, Infineon IRS10x0C and Stereolabs ZED are some examples of this idea, and they are two or three times smaller than the depth cameras presented in this thesis. However, other actors, like Texas Instruments (OPT9221), Hamamatsu (S11963-01CR), Pelican Imaging, Pmd (CamBoard nano) have proposed solutions that can be considered portable. The first example of embedded depth camera has been implemented by the smartphone maker HTC with the One series in 2014. At the same time, Intel is pushing, through the RealSense technology, the depth sensing in laptops and tablets. Finally, also Google with Project Tango is focused into the same target.

This trend represents a common ground with the previous wearable sensors that in the future

will be able to take advantages of the depth information. For example, an activity recognition application for AAL based on "egocentric" vision could improve its performance exploiting a wearable depth camera system.

6.4. Publications

The following articles were published as result of the research carried out during the doctoral study:

Article in scientific journals:

- S. Gasparrini, E. Cippitelli, S. Spinsante, E. Gambi, "A Depth-Based Fall Detection System Using a Kinect Sensor", Sensors 2014, 14(2), 2756-2775;
- 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, 1417-1434.

International conferences:

- E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, "A depth-based joints estimation algorithm for get up and go test using Kinect", 2014 IEEE International Conference on Consumer Electronics (ICCE) in Las Vegas (USA), 10-13 January, 2014;
- E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, "Depth Stream Compression for Enhanced Real Time Fall Detection by Multiple Sensors", IEEE International Conference on Consumer Electronics Berlin (ICCE Berlin), Berlin (DE), 8-11 September, 2014;
- 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" 37^{th} Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Milan (IT), 25-29 August, 2015;
- S. Gasparrini, E. Cippitelli, E. Gambi, S. Spinsante and F. Flórez-Revuelta, "Performance Analysis of Self-Organising Neural Networks Tracking Algorithms for Intake Monitoring Using Kinect", TechAAL 2015, Kingston (UK), 5 November, 2015.

International workshops:

- S. Gasparrini, E. Cippitelli, S. Spinsante, E. Gambi, "Evaluation and Possible Improvements of the ANT Protocol for Home Heart Monitoring Applications", IEEE international workshop on Measurement and Networking, Naples (IT), 7-8 October, 2013;
- 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", International Workshop on Enhanced Living EnvironMENTs (ELEMENT-2014), Wüerzburg (DE), September 24, 2014;

– 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 ICC2015 – Workshop on ICT-Enabled Services and Technologies for eHealth and Ambient Assisted Living, London (UK), 8-12 June, 2015;

– 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, Workshop ELEMENT 2015, to be held in Orhid (FYRO), 1-4 October, 2015.

Book chapters:

– S. Gasparrini, E. Cippitelli, S. Spinsante, E. Gambi, "Depth cameras in AAL environments: technology and real-world applications" in "Assistive Technologies for Physical and Cognitive Disabilities", AMTCP;

– E. Cippitelli, S. Gasparrini, E. Gambi, S. Spinsante, "Quality of Kinect Depth Information for Passive Posture Monitoring" in "Ambient Assisted Living Italian Forum 2013", Springer;

– E. Cippitelli, S. Gasparrini, S. Spinsante, E. Gambi, "Comparison of RGB-D Mapping Solutions for Application to Food Intake Monitoring", in Biosystems & Biorobotics Ambient Assisted Living Italian Forum 2014 Volume 11, Springer.

Appendices

Appendix A.

A.1. Morphological Operations: Erosion, Dilation, Morphological Gradient

These operations are used for different purposes like noise reduction, connect or separate elements, etc. A kernel (or structuring element), used to probe the original image, is centred in the first pixel of the frame and, for each step of the algorithm, it moves from the top left coordinates to the bottom right. Its shape and size (support) are variable and they are typically symmetric around the central point, called *anchor*. Different patterns (disk, hexagon, square, line segment, diamond) can be selected taking into account the characteristics of the input image, but usually the square shape is the classical choice.

In details, the morphological operations are performed between the kernel and the corresponding portion of the original frame. For each step of the algorithm, the partial result is saved in a different frame from the input image. The elementary operations are *dilation* and *erosion*.

$$\begin{aligned} dilate : F(u,v) &= \max_{(i,j)\in kernel} F(u+i, v+i) \\ erode : F(u,v) &= \min_{(i,j)\in kernel} F(u+i, v+i) \end{aligned} \tag{A.1}$$

In the *dilation* the maximum value inside the kernel substitutes the pixel in the *anchor* position, whereas for the erosion the minimum value is taken. The result of the *dilation* (*erosion*) operation is that the regions with high values grow (reduce) in dimension. In particular, the *dilation* smooths concavities and it is used to merge together two blobs, with similar intensity characteristics, that have been previously separated by some noise superimposed to the original image. On the other hand, the *erosion* removes the protrusions and emphasizes the holes (elements with low values). Usually, the input of these two operations is a binary frame, but they can be used also with grayscale and RGB images or depth frames.

The previous algorithms cannot be applied for the pixels in the image's borders. Indeed, in those situations there are some empty entries in the kernel. One solution is to create a virtual border outside the image copying the same pixels in the original border. This idea is repeated for the four borders of the image.

The *opening* and *closing* are two additional operations obtained from the combination of *erosion* and *dilation*. In particular, starting from the original image the *erosion* is performed before the dilation, the result is the *opening*. For the *closing* the order is inverted. The result for the *opening*

(*closing*) is similar to *erode* (*dilate*), but the area of the blobs is less altered.

Another useful operation is the *morphological gradient*:

$$MorpGrad(F) = dilate(F) - erode(F) \tag{A.2}$$

It is equal to the difference between the output of *dilation* and *erosion*. The result, when used with a binary image is to highlight the blob's borders.

A.2. Edge Detection Algorithms

The image borders are pixels characterized by fast changes in brightness levels. Therefore, to identify these points, the 2D derivatives are calculated. This operation is a continuous function, but due to the discrete space where these operations are performed (pixels), the derivatives are approximated with differences between pixel values. Partial derivatives are calculated for different directions, than the results are combined to find the gradient for each edge. Since these operations are shift-invariant and independent, they are implemented using the convolution. In particular, for the *Sobel* algorithm the following kernels are convolved with the input image:

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{A.3}$$

Then, for each pixel the magnitude of the gradient is approximated with:

$$|G(u,v)| = |G_x(u,v)| + |G_y(u,v)| \qquad G_x = h_x * F \quad G_y = h_y * F \tag{A.4}$$

The pixel (u, v) is classified as an edge point if its $G(u, v)$ is greater than a specific threshold. With a kernel greater than a 3×3 matrix, the number of operations increase, but the approximation of the derivative is improved and the result is less influenced by the noise. However, taking into account that the derivatives are in the x and y directions, there is still an inaccuracy for the other directions. Alternative kernels used for the first derivative are the *Scharr*, *Kirsch Compass*, *Prewitt* and *Roberts* kernels.

Usually, the edge transitions are not sharp, therefore it is difficult to find the precise location of the border. The second derivative edge detection algorithm uses the *Laplacian* kernel to overcome this problem. In the discrete space, two possible representations are:

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & -2 \end{bmatrix} \tag{A.5}$$

These kernels are useful to find isolated points in the image that are surrounded by other pixels with different intensity values. The central element in the two matrices has a high value compared

to the other, therefore it leads to an edge detection solution influenced by noise. To limit this issue a smooth solution precedes this algorithm. Compared to the first derivatives, the *Laplacian* kernel gives in output the precise coordinates of the edge.

The *Canny* edge detection algorithm uses the first and second derivatives to provide in output the magnitude of the intensity gradient.

A.3. Hough Transform - Linear and Circular version

Differently from the previous operations, where the characteristics of the images are altered, here the Hough transform [177] changes the domain. In particular, from pixel space to an alternative one where the research of elementary object is easier. Two versions of the Hough transform are described: the first one is used to find straight lines, while the second one to seek circular objects.

It is worth noting that the same elements can be found using alternative solutions, for example exploiting morphological operations with a linear segment as kernel if the object to seek is a line. However, the line orientation and the dimension are usually unknown, therefore different tests must be implemented. The Hough transform overcomes these limits if some characteristics of the object are specified.

The algorithm needs in input the object's borders to work in a proper manner. Therefore, when a real image is given in input to the Hough transform, an edge detection algorithm is often used as pre-processing step.

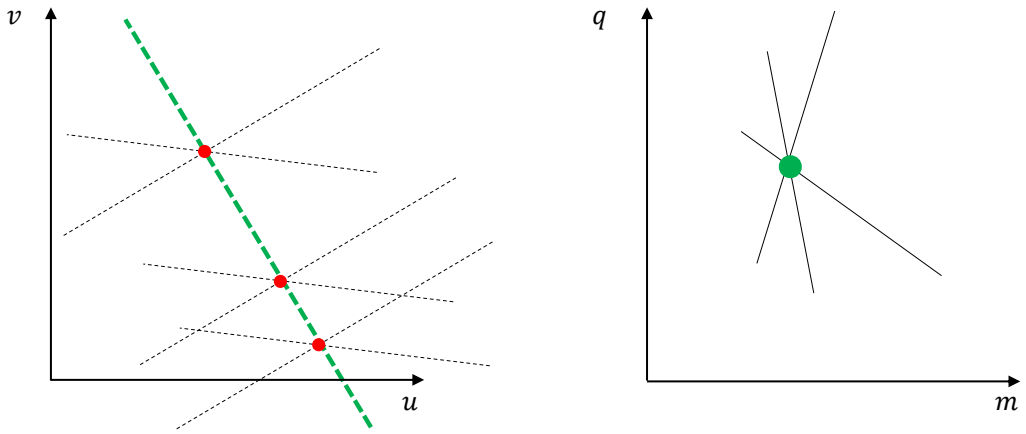
In Figure A.1 is visible the frame domain with three pixels in foreground highlighted in red. The purpose of the Hough function is to find the equation parameters of the green line that connects the three pixels. Each line can be represented by its slope (m) and intercept (q). As sketched in Figure A.1a, infinite lines pass through each pixel. Rewriting the line equation and considering (u, v) as parameters and (m, q) as variables (Eq. (A.6)), the infinite lines are mapped in a single line in the (m, q) domain. Indeed, as visible in Figure A.1b, for each point in the (u, v) plane there is a line in (m, q) space. The intersection point, highlighted in green, is the line to find.

$$v = mu + q \qquad\qquad q = -mu + v \tag{A.6}$$

In the discrete domain, the space (m, q) is divided in cells and renamed *accumulator plane*. When a point from (u, v) is mapped to (m, q), only some of these cells are incremented. In practical terms, setting a discrete range for m and knowing (u, v), all the corresponding q values are calculated.

This operation is repeated for all the foreground pixels, and at the end, the point in (m, q) domain with the highest value is the line requested. Ideally, the green point will have a value equal to 3 but in practice, due to possible noise that shifts the pixels from the straight line, it is represented by a cluster of points with high values close to the green one.

The real algorithm that implements this idea does not use the (m, q) representation, but the polar coordinates (θ, ρ). ρ is the orthogonal distance from the origin to the line and θ is the angle between the u-axis and the orthogonal segment to the line. In this way, some critical problems



(a) (b)

Figure A.1.: Frame domain (a) and Hough domain (b).

such as infinite value for m when the line is vertical or no limited ranges for m and q, are overcome.

$$u\,cos(\theta) + v\,sin(\theta) = \rho \tag{A.7}$$

In the *accumulator plane*, the lines are substituted with curves that, also in this situation, intersect in a single point. The range of $\theta \in \pm 90^{\circ}$ and the limit for ρ can be directly related to the frame dimensions.

When the element to find is a circle, it can be represented with the following equation:

$$(u-a)^2 + (v-b)^2 = c^2 \tag{A.8}$$

where (a, b) is the central point of the object and c is its radius. Therefore, compared with the previous version of the Hough transform, now the number of unknowns is increased. It means that the *accumulator plane* in place of 2D domain will be a 3D space (*accumulator volume*). However, if the radius of the circle to find is known, the number of unknowns goes back to two.

This algorithm can be extended to all the shapes that have a parametric description, adding dimensions to the accumulator domain. Unfortunately, the most important drawback is that the computational complexity rapidly grows when the object to find is represented by many parameters. A possible optimization, to partially deal with this problem, is to initially process the image with a low resolution to find the local maximum in the accumulator space. Then the operation is repeated using the standard resolution close to the local maximum to improve the result.

Bibliography

[1] "Definition of an older or elderly person," http://www.who.int/healthinfo/survey/ageingdefnolder/en/, accessed: 2016-01-21.

[2] P. Laslett, "The emergence of the third age," *Ageing and Society*, vol. 7, no. 2, pp. 133–160, 1987.

[3] UN, *World Population Prospects (The 2015 Revision)*. New York: United Nations, 2015.

[4] P. Parc, "Special Issue on Ambient Assisted Living," *J. Intell. Syst.*, vol. 24, no. 3, p. 299–300, 2015.

[5] E. C. D. G. for Economic and F. Affairs, *The 2015 Ageing Report, Economic and budgetary projections for the 28 EU Member States (2013-2060)*. Publications Office of the European Union, 2015.

[6] M. Cabrera and R. Özcivelek, "ICT for independent living services," *Information and Communication Technologies for Active Ageing: Opportunities and Challenges for the European Union*, vol. 23, pp. 216–234, 2009.

[7] P. Toumpaniaris, D. Iliopoulou, A. Lazakidou, N. Katevas, and D. Koutsouris, "A survey for chronic diseases management and the related sensors in the Ambient Assisted Living environment," *International Journal of Health Research and Innovation*, vol. 2, no. 1, pp. 65–84, 2014.

[8] S. Shaikh and H. Verma, "Parkinson's disease and anaesthesia," *Indian Journal of Anaesthesia*, vol. 55, no. 3, pp. 228–234, 2011.

[9] P. Rashidi and A. Mihailidis, "A survey on Ambient-Assisted Living tools for older adults," *Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[10] G. Acampora, D. Cook, P. Rashidi, and A. Vasilakos, "A survey on Ambient Intelligence in healthcare," in *Proceedings of the IEEE*. IEEE, Dec 2013.

[11] M. Memon, S. Wagner, C. Pedersen, F. Beevi, and F. Hansen, "Ambient Assisted Living healthcare frameworks, platforms, standards, and quality attributes," *Sensors*, vol. 14, pp. 4312–4341, 2014.

[12] A. Kung and B. Jean-Bart, "Making AAL platforms a reality," in *Proceedings of the First International Joint Conference on Ambient Intelligence*. Springer-Verlag, 2010, pp. 187–196.

[13] H. Gokalp and M. Clarke, "Monitoring activities of daily living of the elderly and the potential for its use in telecare and telehealth: a review," *Telemedicine journal and e-health*, vol. 19, no. 12, pp. 910–923, 2013.

[14] M. Abo-Zahhad, S. M. Ahmed, and O. Elnahas, "A wireless emergency telemedicine system for patients monitoring and diagnosis," *Int. J. Telemedicine Appl.*, 2014.

[15] P. Brauner, A. Holzinger, and M. Ziefle, "Ubiquitous Computing at its best: Serious exercise games for older adults in Ambient Assisted Living environments; a technology acceptance perspective," *EAI endorsed transactions on serious games*, vol. 15, no. 4, 2015.

[16] A. Hwang, K. Truong, J. Cameron, E. Lindqvist, L. Nygård, and A. Mihailidis, "Co-designing Ambient Assisted Living (AAL) environments: Unravelling the situated context of informal dementia care," *BioMed Research International*, 2015.

[17] E. Torta, J. Oberzaucher, F. Werner, R. Cuijpers, and J. Juola, "Attitudes towards socially assistive robots in intelligent homes: Results from laboratory studies and field trials," *Journal of Human-Robot Interaction (JHRI)*, vol. 1, no. 2, 2012.

[18] L. Pigini, D. Facal, L. Blasi, and R. Andrich, "Tele-operated robots in elderly care at home: A survey on needs and perceptions of elderly people and caregivers," *Assistive Technology Research Series*, vol. 29, pp. 542–549, 2011.

[19] EU, *Final Evaluation of the Ambient Assisted Living Joint Programme.* European Union, 2013.

[20] K. Marcus, *AAL Project success stories.* Insight Publishers Ltd., 2013.

[21] "Policies for ageing well with ICT," http://ec.europa.eu/digital-agenda/en/policies-ageing-well-ict, accessed: 2016-01-21.

[22] B. M. Lynch, *The Silver Dollar – Longevity Revolution Primer.* BofA Merrill Lynch, 2014.

[23] J. Padilla-López, A. Chaaraoui, and F. Flórez-Revuelta, "Visual privacy protection methods: A survey," *Expert Syst. Appl.*, vol. 42, no. 9, pp. 4177–4195, 2015.

[24] H. Hawley-Hague, E. Boulton, A. Hall, K. Pfeiffer, and C. Todd, "Older adults' perceptions of technologies aimed at falls prevention, detection or monitoring: A systematic review," *International Journal of Medical Informatics*, vol. 83, no. 6, pp. 416–426, 2014.

[25] U. Laessoe, H. Hoeck, O. Simonsen, T. Sinkjaer, and M. Voigt, "Fall risk in an active elderly population – can it be assessed?" *Journal of Negative Results in BioMedicine*, vol. 6, no. 1, pp. 1–11, 2007.

[26] W. H. Organization, *WHO global report on falls prevention in older age.* WHO, 2008.

[27] F. Bianchi, S. Redmond, M. Narayanan, S. Cerutti, and N. Lovell, "Barometric pressure and triaxial accelerometry-based falls event detection," *Neural Systems and Rehabilitation Engineering*, vol. 18, no. 6, pp. 619–627, 2010.

[28] N. Pannurat, S. Thiemjarus, and E. Nantajeewarawat, "Automatic fall monitoring: A review," *Sensors*, vol. 14, no. 7, 2014.

[29] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *BioMedical Engineering OnLine*, vol. 12, no. 1, 2013.

[30] A. Bourke, J. O'Brien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait and Posture*, vol. 26, no. 2, pp. 194–199, 2007.

[31] "Safety for seniors," www.eurosafe.eu.com/csi/eurosafe2006.nsf/wwwVwContent/l2safetyforseniors-seniornew.htm, accessed: 2016-01-21.

[32] M. Kangas, I. Vikman, J. Wiklander, P. Lindgren, L. Nyberg, and T. Jämsä, "Sensitivity and specificity of fall detection in people aged 40 years and over," *Gait and Posture*, vol. 29, no. 4, pp. 571–574, 2009.

[33] A. Özdemir and B. Barshan, "Detecting falls with wearable sensors using machine learning techniques," *Sensors*, vol. 14, no. 6, 2014.

[34] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait and Posture*, vol. 28, pp. 285–291, 2008.

[35] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls," *PloS one*, vol. 7, no. 5, 2012.

[36] T. Banerjee, J. Keller, M. Skubic, and E. Stone, "Day or night activity recognition from video using fuzzy clustering techniques," *Fuzzy Systems, IEEE Transactions on*, vol. 22, no. 3, pp. 483–493, 2014.

[37] H. Nait-Charif and S. McKenna, "Activity summarisation and fall detection in a supportive home environment," in *Pattern Recognition, 2004.* IEEE, Aug 2004.

[38] R. Cucchiara, A. Prati, and R. Vezzani, "A multi-camera vision system for fall detection and alarm generation," *Expert Systems*, vol. 24, no. 5, pp. 334–345, 2007.

[39] A. Leone, G. Diraco, and P. Siciliano, "Detecting falls with 3D range camera in Ambient Assisted Living applications: A preliminary study," *Medical Engineering and Physics*, vol. 33, pp. 770–781, 2011.

[40] E. Stone and M. Skubic, "Fall detection in homes of older adults using the Microsoft Kinect," *Biomedical and Health Informatics*, vol. 19, no. 1, pp. 290–301, 2015.

[41] 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*. Springer Berlin Heidelberg, 2011, pp. 121–128.

[42] Z. Zhang, C. Conly, and V. Athitsos, "Evaluating depth-based computer vision methods for fall detection under occlusions," in *Advances in Visual Computing*. Springer International Publishing, 2011, pp. 196–207.

[43] M. Kepski and B. Kwolek, "Fall detection using ceiling-mounted 3D depth camera," in *Computer Vision Theory and Applications (VISAPP)*, vol. 2. IEEE, Jan 2014, pp. 640–647.

[44] D. Liciotti, G. Massi, E. Frontoni, A. Mancini, and P. Zingaretti, "Human activity analysis for in-home fall risk assessment," in *Communication Workshop (ICCW)*. IEEE, June 2015, pp. 284–289.

[45] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from a single depth image," in *CVPR*. IEEE, June 2011.

[46] "OpenNI 2 SDK binaries and docs," http://structure.io/openni, accessed: 2016-01-21.

[47] G. Mastorakis and D. Makris, "Fall detection system using Kinect's infrared sensor," *J. Real-Time Image Process.*, vol. 9, no. 4, pp. 635–646, 2014.

[48] 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.

[49] G. Diraco, A. Leone, P. Siciliano, M. Grassi, and P. Malcovati, "A multi-sensor system for fall detection in Ambient Assisted Living contexts," in *SENSORNETS*. SciTePress, 2012, pp. 213–219.

[50] B. Kwolek and M. Kepski, "Improving fall detection by the use of depth sensor and accelerometer," *Neurocomput.*, vol. 168, pp. 637–645, 2015.

[51] H. Shimada, M. Suzukawa, A. Tiedemann, K. Kobayashi, H. Yoshida, and T. Suzuki, "Which neuromuscular or cognitive test is the optimal screening tool to predict falls in frail community-dwelling older people?" *Gerontology*, vol. 55, pp. 532–538, 2009.

[52] P. Cheng, M. Liaw, M. Wong, F. Tang, M. Lee, and P. Lin, "The sit-to-stand movement in stroke patients and its correlation with falling," *Archives of Physical Medicine and Rehabilitation*, vol. 79, pp. 1043–1046, 1998.

[53] M. Mak and C. Hui-Chan, "The speed of sit-to-stand can be modulated in Parkinson's disease," *Clinical Neurophysiology*, vol. 116, no. 4, pp. 780–789, 2005.

[54] Q. An, Y. Ishikawa, J. Nakagawa, A. Kuroda, H. Oka, H. Yamakawa, A. Yamashita, and H. Asama, "Evaluation of wearable gyroscope and accelerometer sensor (pocketIMU2) during walking and sit-to-stand motions," in *RO-MAN*. IEEE, Sept 2012, pp. 731–736.

[55] Y. Cedervall, K. Halvorsen, and A. Åberg, "A longitudinal study of gait function and characteristics of gait disturbance in individuals with Alzheimer's disease," *Gait and Posture*, vol. 39, no. 39, pp. 1022–1027, 2014.

[56] F. Wang, M. Skubic, C. Abbott, and J. Keller, "Quantitative analysis of 180 degree turns for fall risk assessment using video sensors," in *EMBC*. IEEE, Aug 2011, pp. 7606–7609.

[57] B. Galna, G. Barry, D. Jackson, D. Mhiripiri, P. Olivier, and L. Rochester, "Accuracy of the Microsoft Kinect sensor for measuring movement in people with Parkinson's disease," *Gait and Posture*, vol. 39, pp. 1062–1068, 2014.

[58] O. Lohmann, T. Luhmann, and A. Hein, "Skeleton Timed Up and Go," in *BIBM*. IEEE, Oct 2012.

[59] B. Kargar, A. Mollahosseini, T. Struemph, W. Pace, R. Nielsen, and M. Mahoor, "Automatic measurement of physical mobility in Get-Up-and-Go Test using Kinect sensor," in *EMBC*. IEEE, Aug 2014.

[60] "EuroDish: Study the need for food and health research infrastructures in Europe," http://www.eurodish.eu/, accessed: 2016-01-21.

[61] EC, *Food security, sustainable agriculture and forestry, marine and maritime and inland water research and the bioeconomy*. European Commission, 2015.

[62] K. Scales and J. Pilsworth, "The importance of fluid balance in clinical practice," *Nursing Standard*, vol. 22, no. 47, pp. 50–57, 2008.

[63] G. Howard, J. Bartram *et al.*, *Domestic water quantity, service level, and health*. World Health Organization Geneva, 2003.

[64] P. Phillips, B. Rolls, J. Ledingham, M. Forsling, J. Morton, M. Crowe, and L. Wollner, "Reduced thirst after water deprivation in healthy elderly men," *N. Engl. J. Med.*, vol. 311, no. 12, pp. 753–759, 1984.

[65] J. Baxter, "Disease-related malnutrition: an evidence-based approach to treatment," *Journal of Human Nutrition and Dietetics*, vol. 16, no. 5, 2003.

[66] T. Andreyeva, P. Michaud, and A. Van Soest, "Obesity and health in europeans aged 50 years and older," *Public Health*, vol. 121, no. 7, pp. 497–509, 2007.

[67] "Obesity and overweight," http://who.int/mediacentre/factsheets/fs311/en, accessed: 2016-01-21.

[68] W. H. Organization, *Obesity: preventing and managing the global epidemic*. World Health Organization, 2000.

[69] A. Martone, G. Onder, D. Vetrano, E. Ortolani, M. Tosato, E. Marzetti, and F. Landi, "Anorexia of aging: A modifiable risk factor for frailty," *Nutrients*, vol. 5, no. 10, pp. 4126–4133, 2013.

[70] J. Wurtman, H. Lieberman, R. Tsay, T. Nader, and B. Chew, "Calorie and nutrient intakes of elderly and young subjects measured under identical conditions," *Journal of Gerontology*, vol. 43, no. 6, pp. B174–B180, 1988.

[71] D. Schoeller, "Limitations in the assessment of dietary energy intake by self-report," *Metabolism*, vol. 44, pp. 18–22, 1995.

[72] A. Adams, S. Soumerai, J. Lomas, and D. Ross-Degnan, "Evidence of self-report bias in assessing adherence to guidelines," *International Journal for Quality in Health Care*, vol. 11, no. 3, pp. 187–192, 1999.

[73] J. Reid, E. Robb, D. Stone, P. Bowen, R. Baker, S. Irving, and M. Waller, "Improving the monitoring and assessment of fluid balance," *Nurs. Times.*, vol. 100, pp. 36–39, 2004.

[74] K. Chang, S. Liu, H. Chu, J. Hsu, C. Chen, T. Lin, C. Chen, and P. Huang, "The diet-aware dining table: Observing dietary behaviors over a tabletop surface," *Pervasive Computing*, vol. 3968, pp. 366–382, 2006.

[75] S. Passler and W. Fischer, "Food intake monitoring: Automated chew event detection in chewing sounds," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 1, pp. 278–289, 2014.

[76] A. Woda, K. Foster, A. Mishellany, and M. Peyron, "Adaptation of healthy mastication to factors pertaining to the individual or to the food," *Physiology and Behavior*, vol. 89, no. 1, pp. 28–35, 2006.

[77] T. Olubanjo and M. Ghovanloo, "Real-time swallowing detection based on tracheal acoustics," in *ICASSP*, May 2014, pp. 4384–4388.

[78] Y. Dong, J. Scisco, M. Wilson, E. Muth, and A. Hoover, "Detecting periods of eating during free-living by tracking wrist motion," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1253–1260, 2014.

[79] E. Mendi, O. Ozyavuz, E. Pekesen, and C. Bayrak, "Food intake monitoring system for mobile devices," in *Advances in Sensors and Interfaces*, June 2013, pp. 31–33.

[80] E. Thomaz, I. Essa, and G. Abowd, "A practical approach for recognizing eating moments with wrist-mounted inertial sensing," in *International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 1029–1040.

[81] Z. Fengqing, M. Bosch, N. Khanna, C. Boushey, and E. Delp, "Multiple hypotheses image segmentation and classification with application to dietary assessment," *Biomedical and Health Informatics*, vol. 19, no. 1, pp. 377–388, 2015.

[82] Y. He, N. Khanna, C. Boushey, and E. Delp, "Image segmentation for image-based dietary assessment: A comparative study," in *ISSCS*. IEEE, July 2013.

[83] P. Pouladzadeh, S. Shirmohammadi, and T. Arici, "Intelligent SVM based food intake measurement system," in *CIVEMSA*. IEEE, July 2013, pp. 87–92.

[84] J. Dehais, S. Shevchik, P. Diem, and S. Mougiakakou, "Food volume computation for self dietary assessment applications," in *Bioinformatics and Bioengineering*. IEEE, Nov 2013, pp. 1–4.

[85] T. Maekawa, "A sensor device for automatic food lifelogging that is embedded in home ceiling light: A preliminary investigation," in *Pervasive Computing Technologies for Healthcare*. IEEE, May 2013, pp. 405–407.

[86] A. Iosifidis, E. Marami, A. Tefas, and I. Pitas, "Eating and drinking activity recognition based on discriminant analysis of fuzzy distances and activity volumes," in *ICASSP*. IEEE, March 2012, pp. 2201–2204.

[87] A. Cunha, L. Pádua, L. Costa, and P. Trigueiros, "Evaluation of MS Kinect for elderly meal intake monitoring," *Procedia Technology*, vol. 16, pp. 1383–1390, 2014.

[88] J. Tham, Y. Chang, and M. Ahmad Fauzi, "Automatic identification of drinking activities at home using depth data from RGB-D camera," in *ICCAIS*. IEEE, Dec 2014, pp. 153–158.

[89] H. M. Hondori, M. Khademi, and C. Lopes, "Monitoring intake gestures using sensor fusion (Microsoft Kinect and inertial sensors) for smart home tele-rehab setting," in *Engineering in Medicine and Biology Society Conference on Healthcare Innovation*. IEEE, Nov 2012.

[90] R. Burgess, T. Hartley, Q. Mehdi, and R. Mehdi, "Monitoring of patient fluid intake using the Xbox Kinect," in *CGAMES*. IEEE, July 2013, pp. 60–64.

[91] J. Lei, X. Ren, and D. Fox, "Fine-grained kitchen activity recognition using RGB-D," in *Conf. Ubiquitous Comput.* ACM, 2012, pp. 208–211.

[92] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *CVPR*. IEEE, 1997, pp. 1106–1112.

[93] A. Bradski, *Learning OpenCV, (Computer Vision with OpenCV Library; software that sees)*, 1st ed. O'Reilly Media, 2008.

[94] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision Second Edition*. Cambridge University Press, 2004.

[95] C. Mutto, P. Zanuttigh, and G. Cortelazzo, *Time-of-Flight Cameras and Microsoft Kinect*. Springer Publishing Company, Incorporated, 2012.

[96] J. Salvi, J. Pagès, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, pp. 827–849, 2004.

[97] L. Zhang, B. Curless, and S. Seitz, "Rapid shape acquisition using color structured light and multi-pass dynamic programming," in *3D Data Processing, Visualization, and Transmission*. IEEE, 2002, pp. 24–36.

[98] ——, "Spacetime stereo: Shape recovery for dynamic scenes," in *CVPR*. IEEE, 2003, pp. 367–374.

[99] L. Li, *Time-of-Flight Camera – An Introduction*. Texas Instruments, 2014.

[100] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight Kinect," *Computer Vision and Image Understanding*, vol. 139, p. 1–20, 2015.

[101] R. Lange, "3D time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology," Ph.D. dissertation, University of Siegen, 2000. [Online]. Available: http://d-nb.info/960293825/34

[102] B. Büttgen, T. Oggier, M. Lehmann, R. Kaufmann, and F. Lustenberger, "CCD/CMOS lock-in pixel for range imaging: challenges, limitations and state-of-the-art," in *Range Imaging Research Day*. CSEM, 2005, pp. 21–32.

[103] R. Un and W. Zhao, "A survey of applications and human motion recognition with Microsoft Kinect," *IJPRAI*, vol. 29, 2015.

[104] Z. Zalevsky, A. Shpunt, A. Malzels, and J. Garcia, "Method and system for object reconstruction," US Patent US 8 400 494 B2, Oct 14, 2006.

[105] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," US Patent US 20 100 118 123 A1, May 13, 2008.

[106] K. Khoshelham and S. Elberink, "Accuracy and resolution of Kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, 2012.

[107] "Kinect for Windows Runtime v1.8," http://www.microsoft.com/en-us/download/details.aspx?id=40277, accessed: 2016-01-21.

[108] "Drivers and libraries for the Xbox Kinect device," https://github.com/OpenKinect/libfreenect, accessed: 2016-01-21.

[109] "Tracking users with Kinect skeletal tracking," https://msdn.microsoft.com/en-us/library/jj131025.aspx, accessed: 2016-01-21.

[110] A. Payne, A. Daniel, A. Mehta, B. Thompson, C. Bamji, D. Snow, H. Oshima, L. Prather, M. Fenton, L. Kordus, P. O'Connor, R. McCauley, S. Nayak, S. Acharya, S. Mehta, T. Elkhatib, T. Meyer, T. O'Dwyer, T. Perry, V.-H. Chan, V. Wong, V. Mogallapu, W. Qian, and Z. Xu, "A 512x424 CMOS 3D time-of-flight image sensor with multi-frequency photo-demodulation up to 130MHz and 2GS/s DC," in *SSCC*. IEEE, 2014, pp. 134–135.

[111] T. Breuer, C. Bodensteiner, and M. Arens, "Low-cost commodity depth sensor comparison and accuracy analysis," in *SPIE*, 2014.

[112] Z. Xu, T. Perry, and G. Hills, "Method and system for multi-phase dynamic calibration of three-dimensional (3D) sensors in a time-of-flight system," US Patent US 8 587 771 B2, Nov 19, 2013.

[113] J. Sell and P. O'Connor, "The Xbox one system on a chip and Kinect sensor," *Micro, IEEE*, vol. 34, no. 2, pp. 44–53, 2014.

[114] "Kinect for Windows SDK 2.0," http://www.microsoft.com/en-us/download/details.aspx?id=44561, accessed: 2016-01-21.

[115] "Open source drivers for the Kinect for Windows v2 device," https://github.com/OpenKinect/libfreenect2, accessed: 2016-01-21.

[116] A. Maykol Pinto, P. Costa, A. Moreira, L. Rocha, G. Veiga, and E. Moreira, "Evaluation of depth sensors for robotic applications," in *ICARSC*. IEEE, 2015, pp. 139–143.

[117] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart, "Kinect v2 for mobile robot navigation: Evaluation and modeling," in *ICAR*. IEEE, July 2015, pp. 388–394.

[118] "JointType Enumeration," https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx, accessed: 2016-01-21.

[119] "SHIMMER," http://www.shimmersensing.com/, accessed: 2016-01-21.

[120] A. Burns, B. Greene, M. McGrath, T. O'Shea, B. Kuris, S. Ayer, F. Stroiescu, and V. Cionca, "SHIMMER A Wireless Sensor Platform for Noninvasive Biomedical Research," *Sensors Journal, IEEE*, vol. 10, no. 9, pp. 1527–1534, 2010.

[121] "MMA7260QT: Three axis low-g Micromachined Accelerometer," https://www.nxp.com/files/sensors/doc/data_sheet/MMA7260QT.pdf, accessed: 2016-01-21.

[122] "Databases for Kinect sensors," http://www.tlc.dii.univpm.it/blog/databases4kinect, accessed: 2016-01-21.

[123] "Kinect Skeleton Stream to BVH Converter," https://kinect2bvh.codeplex.com/, accessed: 2016-01-21.

[124] "Kinect Stream Saver Application SDK 1," http://kinectstreamsaver.codeplex.com/, accessed: 2016-01-21.

[125] "Kinect Version 2 Depth Frame to .mat File Exporter Tool," http://www.codeproject.com/Tips/819613/Kinect-Version-Depth-Frame-to-mat-File-Exporter, accessed: 2016-01-21.

[126] "Kinect-v2-color-frame-recorder," https://github.com/rafaftahsin/Kinect-v2-Color-Frame-Recorder, accessed: 2016-01-21.

[127] "Vitruvius," http://vitruviuskinect.com/ and https://github.com/lightbuzz/vitruvius, accessed: 2016-01-21.

[128] "Kinect Stream Saver Application SDK 2," http://kinectstreamsaver2.codeplex.com/, accessed: 2016-01-21.

[129] "QueryPerformanceCounter function," https://msdn.microsoft.com/it-it/library/windows/desktop/ms644904%28v=vs.85%29.aspx, accessed: 2016-01-21.

[130] "SkeletonFrame.Timestamp Property," https://msdn.microsoft.com/en-us/library/microsoft.kinect.skeletonframe.timestamp.aspx, accessed: 2016-01-21.

[131] "Getting started with SHIMMER streaming to PC," http://www.shimmersensing.com/images/uploads/docs/Streaming_to_PC.pdf and https://www.youtube.com/watch?v=C2UdTdfiQ1g, accessed: 2016-01-21.

[132] "QueryPerformanceFrequency function," https://msdn.microsoft.com/it-it/library/windows/desktop/ms644905%28v=vs.85%29.aspx, accessed: 2016-01-21.

[133] J. Wåhslén, T. Lindh, and M. Eriksson, "A novel approach to multi-sensor data synchronization using mobile phones," in *Fifth International Conference on Body Area Networks*. ACM, July 2010, pp. 171–174.

[134] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," *SIGOPS Oper. Syst. Rev.*, vol. 19, no. 3, pp. 44–54, 1985.

[135] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *ICCV*. IEEE, 1999, pp. 666–673.

[136] ——, "A flexible new technique for camera calibration," *Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

[137] D. Brown, "Close-range camera calibration," *Photogrammetric Engineering and Remote Sensing*, vol. 37, no. 8, pp. 855–866, 1971.

[138] "NuiImageGetColorPixelCoordinatesFromDepthPixelAtResolution," https://msdn.microsoft.com/en-us/library/jj663858.aspx, accessed: 2016-01-21.

[139] "NuiImageGetColorPixelCoordinateFrameFromDepthPixelFrameAtResolution Method," https://msdn.microsoft.com/en-us/library/jj663856.aspx, accessed: 2016-01-21.

[140] "CoordinateConverter class," https://github.com/OpenNI/OpenNI2/blob/master/Wrappers/java/OpenNI.java/src/org/openni/CoordinateConverter.java, accessed: 2016-01-21.

[141] "Demo software to visualize, calibrate and process Kinect cameras output," http://rgbdemo.org/index.php/Main/HomePage, accessed: 2016-01-21.

[142] S. Haykin, *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Inc., 2007.

[143] T. Kohonen, *Self-Organizing Maps*. Springer-Verlag Berlin Heidelberg, 2001.

[144] M. Pöllä, T. Honkela, and T. Kohonen, *Bibliography of Self-Organizing Map (SOM) Papers: 2002-2005 Addendum.* Helsinki University of Technology, 2009.

[145] T. Kohonen, "Essentials of the Self-Organizing Map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[146] ——, *MATLAB Implementations and Applications of the Self-Organizing Map*. Unigrafia, 2014.

[147] F. Coleca, A. State, S. Klement, E. Barth, and T. Martinetz, "Self-Organizing Map for hand and full body tracking," *Neurocomputing*, vol. 147, pp. 174–184, 2015.

[148] A. State, F. Coleca, E. Barth, and T. Martinetz, "Hand tracking with an extended Self-Organizing Map," in *Advances in Self-Organizing Maps*. Springer Berlin Heidelberg, 2013, vol. 198, pp. 115–124.

[149] M. Haker, M. Böhme, T. Martinetz, and E. Barth, "Self-Organizing Maps for pose estimation with a time-of-flight camera," in *Dynamic 3D Imaging*. Springer Berlin Heidelberg, 2009, vol. 5742, pp. 142–153.

[150] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, 1995, pp. 625–632.

[151] T. Martinetz and K. Schulten, "A "neural gas" network learns topologies," in *Artificial Neural Networks*, 1991, pp. 397–402.

[152] S. Orts Escolano, "A three-dimensional representation method for noisy point clouds based on growing Self-Organizing Maps accelerated on GPUs," Ph.D. dissertation, University of Alicante, 2014. [Online]. Available: http://hdl.handle.net/10045/36484

[153] J. Holmström, "Growing neural gas experiments with GNG, GNG with utility and supervised GNG," Master's thesis, University of Uppsala, 2002.

[154] S. Marsland, J. Shapiro, and U. Nehmzow, "A Self-Organizing Network that grows when required," *Neural Netw.*, vol. 15, pp. 1041–1058, 2002.

[155] J. García-Rodríguez, A. Angelopoulou, J. M. García-Chamizo, A. Psarrou, S. O. Escolano, and V. M. Giménez, "Autonomous Growing Neural Gas for applications with time constraint: Optimal parameter estimation," *Neural Networks*, vol. 32, pp. 196–208, 2012.

[156] H. Bauer and K. Pawelzik, "Quantifying the neighborhood preservation of Self-Organizing Feature Maps," *IEEE Transactions on Neural Networks*, vol. 3, pp. 570–579, 1992.

[157] F. Flórez, J. García, J. García, and A. Hernández, "Geodesic topographic product: An improvement to measure topology preservation of Self-Organizing Neural Networks," in *Advances in Artificial Intelligence*, 2004, vol. 3315, pp. 841–850.

[158] T. Villmann, R. Der, M. Herrmann, and T. Martinetz, "Topology preservation in Self-Organizing Feature Maps: exact definition and measurement," *IEEE Transactions on Neural Networks*, vol. 8, pp. 256–266, 1997.

[159] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2008.

[160] NASA, Staff of Anthropology Research Project, *Anthropometric Source Book Volume II: A Handbook of Anthropometric Data*, ser. Anthropometric Source Book. NASA, 1978.

[161] W. W. Gay, *Linux Socket Programming: By Example*. Indianapolis, IN, USA: Que Corp., 2000.

[162] F. Pece, J. Kautz, and T. Weyrich, "Adapting standard video codecs for depth streaming," in *EGVE - JVRC*. Eurographics Association, 2011.

[163] G. Wallace, "The JPEG still picture compression standard," in *Communications of the ACM - Special issue on digital multimedia systems*. IEEE, Apr 1991.

[164] M. Kepski, B. Kwolek, and I. Austvoll, "Fuzzy inference-based reliable fall detection using Kinect and accelerometer," in *Artificial Intelligence and Soft Computing*. Springer-Verlag, 2012.

[165] 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, 2013.

[166] E. Papa and A. Cappozzo, "Sit-to-stand motor strategies investigated in able-bodied young and elderly subjects," *Journal of Biomechanics*, vol. 33, no. 9, pp. 1113–1122, 2000.

[167] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun 1981.

[168] M. Klingner, S. Hellbach, M. Kästner, T. Villmann, and H. J. Böhme, "Modeling human movements with Self-Organizing Maps using adaptive metrics," in *In Proceedings of Workshop New Challenges in Neural Computation*, 2012.

[169] F. Coleca, S. Klement, T. Martinetz, and E. Barth, "Real-time skeleton tracking for embedded systems," in *Proc. SPIE*, vol. 8667, 2013.

[170] F. Flórez Revuelta, J. García Chamizo, J. García Rodríguez, and A. Hernández Sáez, "Analysis of the topology preservation of accelerated growing neural gas in the representation of bidimensional objects," in *Current Topics in Artificial Intelligence*, 2004, vol. 3040, pp. 167–176.

[171] J. Serra-Perez, J. Garcia-Rodriguez, S. Orts-Escolano, J. Garcia-Chamizo, J. Montoyo-Bojo, A. Angelopoulou, A. Psarrou, M. Mentzeopoulos, and A. Lewis, "3D gesture recognition with growing neural gas," in *International Joint Conference on Neural Networks*. IEEE, Aug 2013.

[172] B. Fritzke, "A Self-Organizing Network that can follow non-stationary distributions," in *Artificial Neural Networks*. Springer-Verlag, 1997.

[173] A. Bondy and U. Murty, *Graph Theory*. London: Springer-Verlag, 2008.

[174] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, p. 146–160, 1972.

[175] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[176] "Collection and a development kit of MATLAB mex functions for OpenCV library," https://github.com/kyamagu/mexopencv, accessed: 2016-01-21.

[177] R. O. Duda and P. E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, 1972.