Dynamic hand gesture recognition - From traditional handcrafted to recent deep learning approaches

Quentin de Smedt

Année 2017 Num d'ordre: 42562

Université Lille 1

Ecole Doctorale Sciences Pour l'Ingénieur

Laboratoire CRIStAL (umr cnrs 9189)

THÈSE

pour obtenir le grade de

Docteur,

spécialité Informatique

soutenu par

Quentin De Smedt

le 14/12/2017

Dynamic hand gesture recognition - From traditional handcrafted to recent deep learning approaches

COMPOSITION DU JURY

M. Laurent Grisoni	Professeur, Université de Lille 1	Président
Mme. Saida Bouakaz	Professeur, Université Claude Bernard Lyon 1	Rapporteur
M. Fabien Moutarde	Professeur, Mines ParisTech	Rapporteur
M. Francisco Flórez-Revuelta	Associate professor, Université Alicante, Espagne	Examinateur
M. Jean-Philippe Vandeborre	Professeur, IMT Lille Douai	Directeur
M. Hazem Wannous	Maître de conférences, Université de Lille 1	Encadrant

Abstract

Hand gestures are the most natural and intuitive non-verbal communication medium while using a computer, and related research efforts have recently boosted interest. Additionally, data provided by current commercial inexpensive depth cameras can be exploited in various gesture recognition based systems. The area of hand gesture analysis covers hand pose estimation and gesture recognition. Hand pose estimation is considered to be more challenging than other human part estimation due to the small size of the hand, its greater complexity and its important self occlusions. Beside, the development of a precise hand gesture recognition system is also challenging due to high dissimilarities between gestures derived from *ad-hoc*, cultural and/or individual factors of users. First, we propose an original framework to represent hand gestures by using hand shape and motion descriptors computed on 3D hand skeletal features. We use a temporal pyramid to model the dynamic of gestures and a linear SVM to perform the classification. Additionally, we create the Dynamic Hand Gesture dataset containing 2800 sequences of 14 gesture types. Evaluation results show the promising way of using hand skeletal data to perform hand gesture recognition. Experiments are carried out on three hand gesture datasets, containing a set of fine and coarse heterogeneous gestures. Furthermore, results of our approach in terms of latency demonstrated improvements for a low-latency hand gesture recognition systems, where an early classification is needed. Then, we extend the study of hand gesture analysis to online recognition. Using a deep learning approach, we employ a transfer learning strategy to learn hand posture and shape features from depth image dataset originally created for hand pose estimation. Second, we model the temporal variations of the hand poses and its shapes using a recurrent deep learning technology. Finally, both information are merged to perform accurate prior detection and recognition of hand gestures. Experiments on two datasets demonstrate that the proposed approach is capable to detect an occurring gesture and to recognize its type far before its end.

Key words: hand gesture recognition, depth data, skeletal data, deep learning, online detection.

Résumé

Reconnaisance de gestes dynamiques de la main - De la création de descripteurs aux récentes méthodes d'apprentissage profond.

Les gestes de la main sont le moyen de communication non verbal le plus naturel et le plus intuitif lorsqu'il est question d'interaction avec un ordinateur. Les efforts de recherche qui y sont liés ont récemment relancé son intérêt. De plus, les données fournies par des caméras de profondeur actuellement commercialisées à des prix abordables peuvent être exploitées dans une large variété de systèmes de reconnaissance de gestes. L'analyse des gestes de la main s'appuie sur l'estimation de la pose de la main et la reconnaissance de gestes. L'estimation de la pose de la main est considérée comme étant un défi plus important que l'estimation de la pose de n'importe quelle autre partie du corps du fait de la petite taille d'une main, de sa plus grande complexité et de ses nombreuses occultations. Par ailleurs, le développement d'un système précis de reconnaissance des gestes de la main est également difficile du fait des grandes dissimilarités entre les gestes dérivant de facteurs *ad-hoc*, culturels et/ou individuels inhérents aux acteurs. Dans un premier temps, nous proposons un système original pour représenter les gestes de la main en utilisant des descripteurs de forme de main et de mouvement calculés sur des caractéristiques de squelette de main 3D. Nous utilisons une pyramide temporelle pour modéliser la dynamique des gestes et une machine à vecteurs de support (SVM) pour effectuer la classification. De plus, nous proposons une base de données de gestes de mains dynamiques contenant 2800 séquences de 14 types de gestes. Les résultats montrent une utilisation prometteuse des données de squelette pour reconnaître des gestes de la main. Des expérimentations sont menées sur trois ensembles de données, contenant un ensemble de gestes hétérogènes fins et grossiers. En outre, les résultats de notre approche en termes de latence ont démontré que notre système peut reconnaître un geste avant sa fin. Dans un second temps, nous étendons l'étude de l'analyse des gestes de main à une reconnaissance en ligne. En utilisant une approche d'apprentissage profond, nous employons une

strategie de transfert d'apprentissage afin d'entrainer des caractéristiques de pose et de forme de la main à partir d'images de profondeur d'une base de données crée à l'origine pour un problème d'estimation de la pose de la main. Nous modélisons ensuite les variations temporelles des poses de la main et de ses formes grâce à une methode d'apprentissage profond récurrente. Enfin, les deux informations sont fusionnées pour effectuer une détection préalable et une reconnaissance précise des gestes de main. Des expériences menées sur deux ensembles de données ont démontré que l'approche proposée est capable de détecter un geste qui se produit et de reconnaître son type bien avant sa fin.

Mots-clés reconnaisance de gestes de la main, données de profondeur, données de squelettes, apprentissage profond, detection en temps réel.

Acknowledgement

A thesis is often defined as a lonely process made by a single PhD student. In practice, it is the opposite. In hindsight, I never could have reach the end of my thesis without the help and the engagement of many people the last three years.

Special thanks to my PhD committee members for taking the time to participate in this process, and especially the reviewers of the manuscript for having accepted this significant task: Prof. Fabien Moutarde, Prof. Saida Bouakaz, Ass. Prof. Francisco Flórez-Revuelta and Prof. Laurent Grisoni.

I would like to express my gratitude to my advisor, Prof. Jean-Philippe Vandeborre for guiding me through my research with professionalism, understanding, and patience. His high experience strongly helped me to make this experience productive and stimulating.

I also thank my co-advisor, Dr. Hazem Wannous, for his time and advice in beneficial scientific discussions. His friendly and encouraging guidance make me more confident in my research.

A special thanks to my friends and PhD student colleagues with whom I shared this experience: Maxime Devanne, Vincent Léon, Meng Meng, Taleb Alashkar, Matthieu Allon, Vincent Itier, Sarah Ribet, Anis Kacem, Omar Ben Tanfous, Kaouthar Larbi and Nadia Hosni.

I also thank gratefully my family and my friends for their support and encouragements. I would like to cite them all but I am too afraid to forget one and I know they will make me pay for it.

Villeneuve d'Ascq, October 16, 2017.

Publications

International journal

- **Quentin De Smedt**, Hazem Wannous and Jean-Philippe Vandeborre. Heterogeneous Hand Gesture Recognition Using 3D Dynamic Skeletal Data. *Computer Vision and Image Understanding*, (Under review).

International worshops

- **Quentin De Smedt**, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux and David Filliat. SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. *10th Eurographics Workshop on 3D Object Retrieval*, 2017.

- **Quentin De Smedt**, Hazem Wannous and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2016.

- **Quentin De Smedt**, Hazem Wannous and Jean-Philippe Vandeborre. 3D Hand Gesture Recognition by Analysing Set-of-Joints Trajectories. *2nd International Conference on Pattern Recognition (ICPR) Workshops*, 2016.

In progress

- **Quentin De Smedt**, Hazem Wannous and Jean-Philippe Vandeborre. Dynamic gesture recognition using deep hand posture and shape features.

Contents

List of Figures

1 Introduction

Contents

The analysis and the interpretation of human behavior from visual input is one of the most trendy computer vision fields of research. It is not only due to its exciting scientific challenges but also motivated by the increase of societal needs in terms of applications, such as manipulation in real and virtual environments, monitoring systems, health support, security, entertainment, etc. Human behavior analysis from vision cues is composed of sub-domaine of research differing in scale, both, spatially – from face to body analysis – and temporally – from gesture to activity analysis. In this thesis, we focus our research on the analysis and the recognition of gestures based specifically on hand cues.

Among human body parts, hands are the most effective and intuitive interaction tools in Human-Computer Interaction (HCI) applications. In computer vision, the task of hand gesture recognition exists for many years and has attracted many researchers. Last years, the growing interest for virtual and augmented reality applications instigated a real need in accurate hand gesture recognition. Indeed it could allow users to play and interact with a virtual world in the most natural way and offer new endless possibilities of applications.

The area of hand gesture analysis covers hand pose estimation and gesture recognition. Hand pose estimation is considered to be more challenging than other human part estimation due to the small size of the hand, its greater complexity and its important self occlusions. Beside, the development of a precise hand gesture recognition system is also challenging. Different occurrences of the same gesture type contain high dissimilarities derived from *ad-hoc*, cultural and/or individual factors in the style, the position and the speed of gestures. In addition, gestures with different meanings contain high similarities derived from the heterogeneity of possible gestures.

Hand gesture analysis has been widely investigated in the literature, especially from 2D videos captured with RGB cameras. There are, however, challenges confronted by these methods, such as the sensitivity to color and illumination changes, background clutter and occlusions.

The recent release of new effective and low-cost depth sensors allows

researchers to consider the 3D information of the scene and thus easily perform background substraction and hand detection in the scene. In addition, the technology behind such depth sensors is more robust to light variations. Using these depth cameras, researches made available a real-time estimation of 3D humanoid skeletons. Such skeletons are composed of a set of 3D connected joints representing the human body. This data facilitated and improved the analysis of the human pose and its motion over the time. Their effectiveness for human action recognition have motivated very recently researchers to investigate the extraction of 3D skeletal data from depth images to describe precisely the hand pose.

Traditionally, once spatio-temporal hand gesture descriptors have been extracted, machine learning algorithms are used to perform the recognition process. However, recent advances in terms of data and computational resources with powerful hardware lead to a change of paradigm in computer vision, with the uprising of deep learning. Furthermore, motivated by their success for images and videos, researchers are working intensely towards developing models for learning hand pose features. Unprecedented results have been also obtained in many tasks including hand gesture recognition. Not only is it efficient in the recognition task but it also automatically learns relevant descriptors.

1.1 Thesis Contributions

All the above considerations lead this thesis to address the problem of hand gesture recognition, according to two categories of approaches; handcrafted and deep learning. Hence, we investigate in the first part the gesture recognition problem by employing geometric features derived from hand skeleton data, for heterogeneous and fine dynamic hand gestures. The hand pose, can be either captured directly by certain sensors, or extracted later from depth images. A study of the impact of the hand pose estimator on the recognition process will be considered. In the second part, we extend the study to online dynamic hand gestures taking over the whole pipeline of the recognition process, from hand pose estimation

to the classification step, using deep learning. So as to face the main challenges, we propose to revisit the feature pipeline by combining the merits of geometric shape and dynamic appareance, both extracted from a CNN trained for hand pose estimation problem. The main contributions of this thesis can be summarized as follows:

- **Recognition of heterogeneous dynamic hand gesture based on 3D skeletal data:** We propose an original framework to represent hand gesture sequences by using hand shape and motion descriptors computed on 3D hand skeletal features. We use a temporal pyramid to model the dynamic of gestures and a linear SVM to perform the classification. Additionally, prompted by the lack of dynamic hand gesture dataset publicly available providing skeletal features, we create the Dynamic Hand Gesture dataset containing 2800 sequences of 14 gesture types. It allows us to investigate the use of hand skeletal data to recognize heterogeneous grained-fine hand gestures;

- **Online detection and recognition of hand gesture using a deep learning approach:** We extend the study of hand gesture analysis to detection of gestures allowing us to perform online recognition. Using a deep learning approach, we take over the whole pipeline of the hand gesture analysis from the hand pose estimation step to the recognition process. First, we learn hand pose and shape features from depth images. Second, we model the temporal aspect separately of the hand poses and the shape variations over the time using a recurrent deep learning technology. Finally, both information are merged to perform accurate detection and recognition of hand gestures.

1.2 Thesis outline

The thesis is organized as follows: in Chapter 2, we lay out the issues of hand pose estimation and hand gesture recognition from videos as well as existing solutions in the literature. Chapter 3 introduces the algorithm that we employ to analyze and compare the use of skeletal features for dy-

namic hand gesture recognition. Chapter 4 is dedicated to the comparison between traditional and deep learning pipelines as well as introducing the deep learning technology. In Chapter 5, we propose a deep learning framework for detection and recognition of dynamic hand gestures by performing, both, hand posture and shape features extraction and the recognition process. Finally, we conclude the manuscript in Chapter 6 by summarizing the contributions of the thesis and proposing several directions of future research.

Literature overview

2

Contents

2.1 Introduction

In computer vision, the task of hand gesture recognition exists for many years and has attracted many researchers notably because of its wide range of potential applications. In this chapter, we discuss the defined challenges of hand gesture analysis and their potential applications. Then, we introduce the depth sensor technology as well as RGB-D and 3D skeletal data provided by such cameras. We present related works about hand pose estimation problem aiming to retrieve 3D skeletal features from depth images. Benchmark datasets of depth images and/or 3D skeletal data collected for the task of hand gesture recognition are then presented. Finally, we review the main existing state-of-the-art approaches, which provide methodology to tackle the problem of hand gesture recognition.

2.1.1 Hand gesture understanding problem

The field of human motion analysis from vision clues is composed of many sub-areas of research and covers a large variety of tasks including, but not limited to:

- Face expression recognition;
- Hand gesture recognition;
- Upper-body gesture recognition;
- Action recognition;
- Activity recognition;
- Body expression recognition.

They differ in scale, both spatially (i.e. face, hand, upper-body, whole body clues) and temporally (i.e. time to perform an expression, a gesture, an action or yet an activity). Similar approaches can be used to tackle each or a part of those problems. However, each of them has their own particularities that have to be taken into account to create robust and efficient recognition systems.

In this thesis, we focus on hand gesture analysis and recognition based specifically on hand cues. Modeling the deformation of a hand is considered to be more challenging that other human parts due to its smaller size, greater complexity and higher self occlusions. The hand gesture analysis area covers hand modeling and gesture recognition.

First, we focus on hand modeling also called hand pose estimation. This task aims to map an observed input (generally a 2D or a 3D image) to a set of 3D joints together forming a possible hand configuration called hand skeleton or hand pose, which takes into account the anatomic structure constraints of the hand as depicted in Figure 2.1.

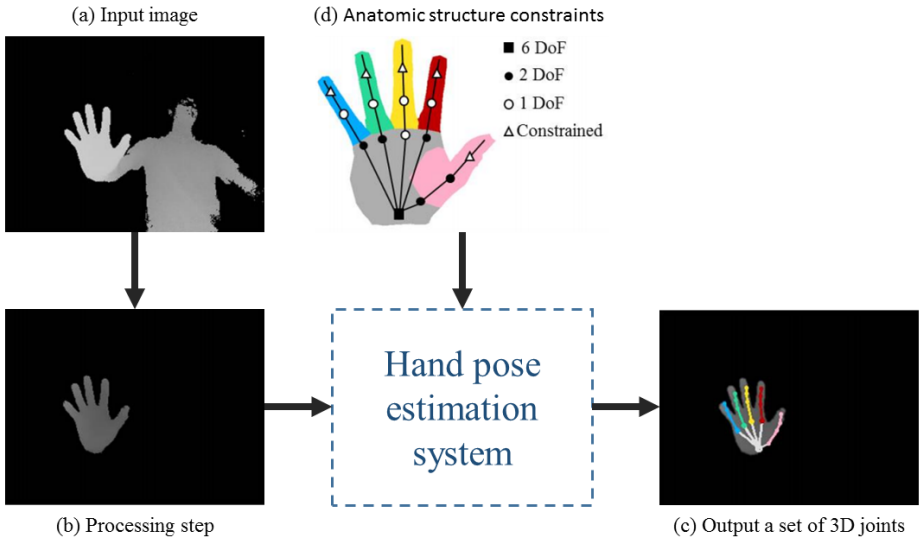


Figure 2.1 – *Illustration of the hand pose estimation task. (a) from an input image and after (b) a pre-processing step, a hand pose estimation system is able to (c) output a set of 3D joints called together hand skeleton or hand pose based on (d) the anatomic structure constraints of the hand.*

Hand skeletal data can further serve as features for the task of gesture recognition. Beside, the task of hand gesture recognition from sequences of images can be defined as follows: given a set of known samples of meaningful hand gesture types from a predefined vocabulary, which type is performed during a new observed sequence? Figure 2.2 illustrates this problem.

The problem can then be extended to the analysis of a long unsegmented stream of gestures, where different hand motions are performed successively and should be recognized and localized in the time by the sys-

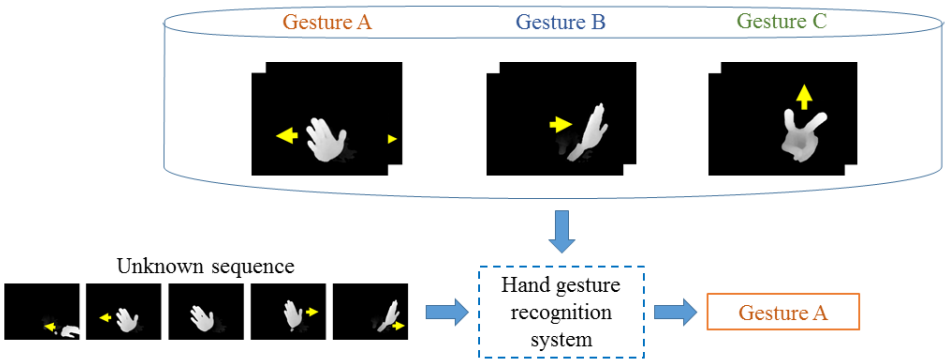


Figure 2.2 – *Illustration of the hand gesture recognition task. Given a set of labeled gesture samples, the system is able to recognize which gesture have been performed in an unknown sequence.*

tem. This task is called *online recognition*. The task of hand gesture recognition can be traditionally (i.e. non deep learning) divided in a pipeline (depicted in Figure 2.3) of ordered sub-tasks as follows:

1. Data generation: creation of a dataset using one or multiple 2D and/or 3D cameras;

2. Preprocessing steps: including localization of the hand region of interest, background subtraction, denoising or yet segmentation;

3. Engineering or representation of data: transforming the input (e.g. an image) in a representation which extracts meaningful features. Most of the time, this task can be realized in several steps as a cascade of successive data transformations;

4. Classification: a machine learning algorithm is used here to help mapping the hand gesture representation to a gesture class.

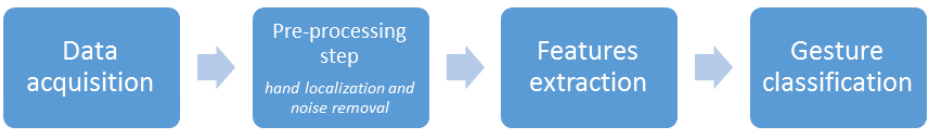


Figure 2.3 – *The pipeline of vision-based traditional hand gesture recognition systems. First, visual data are captured. Second, an optional pre-processing step is often conducted in order to extract the region of interest of the hand from the background. Third, features are computed to extract relevant hand gesture information. Finally, a classification step outputs a label.*

Before the release of new depth sensors, hand gesture recognition has been widely investigated in computer vision from 2D images captured from standard RGB cameras [70, 171, 139, 22]. However, most of these methods suffer of some limitations coming from 2D videos, like the sensitivity to color and illumination changes, background clutter and occlusions.

Since the recent release of RGB-D sensors, like Microsoft Kinect 1 [61] or the Intel RealSense [117], new opportunities have emerged in the field of human motion analysis. Hence, many researchers investigated data provided by such cameras to benefit from their advantages [7]. Indeed, depth data allow to consider the 3D information of the scene and thus easily perform background substraction and detect object in the scene (e.g. human body or hand). In addition, the technologies behind such depth sensors provide more robustness to light variations as well as working in complete darkness.

However, working with human motion analysis poses great challenges. What complicates the task is the necessity of being robust to execution speed and geometric transformations, like the size of the subject, its position in the scene and its orientation with respect to the sensor.

In addition, the task of precise dynamic hand gesture recognition also presents its own challenges. The heterogeneity of possible gesture types arises from a large amount of *intraclass* gesture dissimilarity (coming from ad-hoc, cultural and/or individual factors in the style, position and speed of gestures in addition to the high dexterity of the hand) and *interclass* similarities – derived from high similarities between different type of gestures.

Moreover, an efficient algorithm of hand gesture recognition needs to work with a low computational complexity to be used in real world applications. It should also allow a low latency, making the interaction with the system more fluid.

2.1.2 Applications

The main motivation behind the interest of precise hand gesture recognition, whether using standard cameras or RGB-D sensors, is the large range of applications in various fields, like Human-Computer Interaction, virtual and augmented reality interaction, sign language recognition, and robotics. In many futuristic movies, characters perform human-computer interactions using hand gestures as depicted in Figure 2.4. If current systems do not yet allow such freedom, these examples show the potential impact that interaction systems based on hand gestures could have on the way we interact with computers.

Figure 2.4 – *Two examples of human-computer interaction based on hand gestures: (top) Robert Downey Jr. (Tony Stark) from the movie Iron Man is using his hands to interact with an augmented reality system. (bottom) Tom Cruise (John Anderton) from the movie Minority report interacts with a computer system using hand gestures on a 3D screen.*

A) Interaction with virtual and augmented reality systems

Virtual Reality (VR) is a computer technology that uses headsets to generate images, sounds and other sensations that simulate the user presences in a virtual environment. A person using high quality virtual reality equipment is able to look and move around the artificial world. Augmented reality (AR) is a view of the real-world environment which is *augmented* by computer-generated input. AR enhances the current real

environment, where in contrast, VR replaces the entire real world with a simulated one.

Until recently, VR has been a fantasy for futuristic storytellers. But in 2010, an american teenager, Palmer Luckey, created the first prototype of a VR headset, evolving later into the Oculus Rift [103]. Since, several competitors have emerged, from the HTC Vive [112] and the Sony PlayStation VR [4] to the Samsung Gear VR [124] and the Google Cardboard [85]. Consequently, many developers are developing VR games and applications and filmmakers are investigated the potential for movies. VR and AR technologies has a tremendous amount of potential applications starting from video games but could be also used for art making, education, medical applications, flight training, etc.

Once the user is immersed into the virtual world, it needs to interact with its new environment. Precise hand gesture recognition based on visual cues will allow users to play and interact with a virtual world in the most natural and intuitive way and will offer new endless possibilities of applications as shown as illustrated in Figure 2.5.

Figure 2.5 – *Example of a natural interaction with an augmented reality system using hands.*

B) Sign language recognition

Sign language is the primary language used by people with impaired hearing and speech. People use sign language gestures as a means of

non-verbal communication to express their thoughts and emotions. Generally, sign languages consist of three parts: finger-spelling, word level sign vocabulary, and non-manual features [30]. Finger-spelling is to spell words letter by letter, using the fingers and word level sign vocabulary uses a gesture as a word. Non-manual features include facial expressions, mouth and body poses to indicate meaning.

However, only few people are able to understand sign language and, therefore, impaired hearing people often require the assistance of a trained interpreter. Although, employing an interpreter is expensive and not always available. Robust and efficient sign language recognition system using visual clues could enable cheap, natural and convenient system of interaction with people with impaired hearing and speech.

C) Ambient assisted living

The development of a vision-based assistive system can help patient in their ambient lifestyle by analyzing their daily activities. Indeed, as robots move away from industrial settings and closer into our lives, the question arises of how to simply interact with them in an unconstrained and natural way. A large part of natural interaction happens through hand gestures, especially if a robot is designed to help humans with everyday tasks (e.g. *bring something by pointing at the object*, *go somewhere*, ...). This requires a system that allows the robot to detect an interacting human and, obviously, make it understand hand gestures.

2.2 Acquisition systems of depth images and 3D skeletal data

Analyzing and understanding a real-world scene observed by a camera is the main goal of many computer vision systems. Standard cameras provide only 2D information which have some limitations when analyzing and understanding the scene. Having the full 3D information about the observed scene became an important challenge in computer vision. In a traditional stereo-vision system, using two cameras observing the same

scene from different points of view, we compare the two images to develop a disparity image and estimate the relief of the scene.

For the task of human motion analysis, extracting 3D information about the human pose is a challenge that has attracted many researchers. Motion capture systems are able to capturing an accurate human pose, and track it along the time using markers representing the human pose. Motion capture data have been widely used in industry such as in animation and video games. However, these systems present some disadvantages. First, the high cost of this technology limits its usage. Second, it implies that the subject wears physical markers to estimate the 3D pose.

Recently, new depth sensors have been released, like the Microsoft Kinect [61] or the Asus Xtion [3] PRO LIVE shown in Figure 2.6.

Figure 2.6 – *Example of RGB-D sensors. Left: Microsoft Kinect 2 [61], right: Asus Xtion PRO LIVE [3].*

In addition to standard RGB images, depth sensors provide a depth map giving for each pixel the corresponding distance with respect to the sensor. The 3D information of the scene can be estimated from such depth maps. Behind these depth sensors, there are two types of technology:

- Structured light: a visible or invisible known pattern is projected in the scene. A sensor analyzes the distortion of the pattern in contact with objects and estimates the distance of each point of the pattern;
- Time of flight: a light signal is emitted in the scene. Knowing the speed of light, a receiver computes the distance of the object based on the time elapsed between the emission of the signal and its reception.

Depth sensors, like Microsoft Kinect 1 [61] or Asus Xtion PRO LIVE employ the structured light technique, while the new Microsoft Kinect 2 employs the time of flight. These new acquisition devices have stimulated

the development of various promising applications. A recent review of kinect-based computer vision applications can be found in [47].

In 2011, Shotton *et al.* [134] proposed a real-time method to accurately predict the 3D positions of 20 body joints from single depth image, without using any temporal information. Thus, the human pose can be represented as a 3D humanoid skeleton. Such RGB-D sensors provide for each frame the 2D color image of the scene, its corresponding depth map and a body skeleton representing the subject pose. An example is illustrated in Figure 2.7.

Figure 2.7 – *Example of data provided by RGB-D sensors as Microsoft Kinect 2 [61] : 2D color image (left), depth map (middle) and body 3D skeleton (right).*

Several feature descriptors in the literature proved how the position, the motion, and the orientation of joints could be excellent descriptors for human actions [151, 157, 27].

In the field of hand pose estimation and gesture recognition, many researchers have benefited from 3D information in order to perform reliable and efficient algorithms. The hand is a small and a complex object with a lot of potential self-occlusions and, so, analyzing the hand shape information is very challenging with long range depth cameras like Microsoft Kinect [61] (0.8 - 4.2 meters). Hand gesture analysis does not necessary require information about the whole body, thus, researchers focus recently on data extracted from short range depth cameras as the Intel RealSense SR300 [117] and the SoftKinetic DS325 cameras shown in Figure 2.8. These cameras provide more details from object near the lens of the camera (0.2 - 2 meters) and, so, allow to extract preciser information about the hand shape. A comparison between depth cameras can be found in [53].

In addition to depth images, the software development kit of the Intel

Figure 2.8 – *Example of short range RGB-D sensors. Left: Intel RealSense SR300 [117], right: SoftKinetic DS325 [135].*

RealSense SR300 [117] provides a stream of 3D full hand skeleton of 22 joints at 30 frames per second (see in Figure 2.9). Beside, in july 2013, the Leap Motion Controller (LMC) is launched on the public market. The LMC was primarily designed for hand tracking and provides 3D full hand skeleton of 24 joints. Such data offer new opportunities and axes of research related to hand gesture analysis.

Figure 2.9 – *Example of data provided by short range RGB-D sensors as Intel RealSense [117] : 2D color image (left), depth map (middle) and hand 3D skeleton (right).*

2.3 Datasets for hand pose estimation

The emergence of RGB-D data has encouraged research groups to build new datasets for the task of hand pose estimation from depth images. The current state-of-the-art methods mostly employ deep neural networks to estimate hand pose from a depth image [147, 101, 102, 177, 40, 170]. The availability of a large-scale, accurately annotated dataset is a key factor for advancing this field of research. Consequently, numerous RGB-D datasets have been made publicly available last years. The different hand pose datasets differ in the annotation protocol used, the number of samples, the number of joints in the hand skeleton representation, the view point and

the depth image resolution. Table 2.1 lists the principal current publicly available datasets for hand pose estimation and their characteristics.

Publicly available datasets for evaluation of hand pose estimation algorithms are significantly limited in scale (from a few hundred to tens of thousands samples) and the annotation accuracy is not always efficient. The barrier for building a large-scale dataset based on depth data is the lack of a rapid and accurate annotation method.

Creating a dataset by manual annotation [136, 115] is time consuming and results in inaccurate labels. As a result, these datasets are small in size. MSRA14 [115] and Dexter 1 [136] have only 2,400 and 2,137 samples, respectively, making them unsuitable for large-scale training.

Alternative annotation methods, which are still time consuming, track a hand model and manually refine the results [144, 147, 140]. The ICVL dataset [144] was annotated using 3D skeletal tracking [91] followed by manual refinement. Annotations of the NYU dataset [147] were obtained by model-based hand tracking on depth images from three cameras. These methods often result in incorrect poses where manual correction is needed. The MSRA15 dataset [140] was annotated in an iterative way, where an optimization method [115] and manual refinements were done until convergence. Annotations still contains errors, such as occasionally missing finger annotations.

Two small datasets were made using semi-automatic annotation methods [120, 100]. The UCI-EGO dataset [120] was annotated by searching for the closest example in a synthetic set and manual refinements. In the Graz16 dataset [100] visible joints in a number of key frames were annotated and automatically inferring the complete sequence. Manual refinements are also required when the optimization fails. This semi-automatic method resulted in 2,166 annotated samples which are also insufficient for large-scale training.

Additional sensors can help automatic annotation [164, 161] but attention must be paid to not restrict the naturalness of motion. The ASTAR dataset [164] used a ShapeHand glove [131] depicted in Figure 2.10. How-

Table 2.1 – *Summary of most popular hand pose datasets.*

Dataset	Annotation	No. frames	No. joints	View point	Depth map resolution	Year
Dexter 1 [136]	manual	2,137	5	3rd	320×240	2013
MSRA14 [115]	manual	2,400	21	3rd	320×240	2014
ICVL [144]	track + refine	17,604	16	3rd	320×240	2015
NYU [147]	track + refine	81,009	36	3rd	640×480	2014
MSRA15 [140]	track + refine	76,375	21	3rd	320×240	2015
UCI-EGO [120]	semi-automatic	400	26	ego	320×240	2015
Graz16 [100]	semi-automatic	2,166	21	ego	320×240	2016
ASTAR [164]	automatic	870	20	3rd	320×240	2015
HandNet [161]	automatic	212,928	6	3rd	320×240	2015
MSRC [132]	synthetic	102,000	22	3rd	512×424	2015
BigHand2.2M [172]	automatic	2,2M	21	full	640×480	2017

ever, wearing the glove alter hand depth images and reduce the freedom of motion.

Figure 2.10 – *The ShapeHand [131] glove motion apture systems.*

More recently, less intrusive sensors have been used for finger tip annotation in the HandNet dataset [161] which exploits the trakSTAR magnetic sensors [148]. This dataset only provides fingertip locations, not the full hand annotations.

Synthetic data has also been exploited for generating hand pose data [132]. If an unlimited synthetic data can be generated, it still remains large differences between synthetic and real data.

Currently, the widely used datasets in the literature for benchmarking purposes are the ICVL [144] and the NYU [147] datasets.

Very recently, Yuan *et al.* [172] introduce the million-scale BigHand2.2M dataset and made a significant advancement in terms of the scale and the annotation quality using the same magnetic sensors as Wetzler *et al.* [161] but providing a full hand annotated pose. The dataset contains 2.2 million depth images with 21 accurately annotated joint locations but is not yet publicly available. This large amount of correctly annotated hand pose will allows improvements and new advances in the field of hand pose estimation using depth images.

2.4 Related work on hand pose estimation

Accurate hand pose estimation is an important requirement for many Human-Computer Interaction or Augmented Reality tasks, and has attracted lots of attention in the Computer Vision research community.

2.4.1 Hand pose estimation from RGB images

There has been a significant amount of works that dealt with hand pose estimation using RGB images. Those approaches can be divided into two categories: model based approaches and appearance based approaches [176].

Model based approaches generate hand pose hypotheses and evaluate them with the input image. Heap *et al.* [48] proposed to fit the mesh of a 3D hand model with the surface of the hand by a mesh constructed via Principal Component Analysis (PCA) from training samples. Real-time tracking is achieved by finding the closest possibly deformed model matching the image. Henia et al. [50] used a two-step minimization algorithm for model-based hand tracking. They proposed a new dissimilarity function and a minimization process that operates in two steps: the first one provides the global parameters of the hand, i.e. position and orientation of the palm, whereas the second step gives the local parameters of the hand, i.e. finger joint angles. However, those methods are unable to handle the occlusion problem.

Appearance based methods use directly the information contained in images. They do not use an explicit prior model of the hand, but rather seek to extract the region of interest of the hand in the image. Bretzner *et al.* [13] used color features to recognize hand shapes. Therefore, the hand can be described as one big blob feature for the palm, having smaller blob features representing fingers. This became a very popular method but has some drawbacks such as skin color detection which is very sensitive to lighting conditions. We refer the reader to Garg *et al.* [39] for an overview of hand pose estimation based on RGB approaches.

2.4.2 Hand pose estimation from depth images

The hand pose estimation community has rapidly grown larger in recent years. The introduction of commodity depth sensors and the multitude of potential applications have stimulated new advances. However, it is still challenging to achieve efficient and robust estimation performance because of large possible variations of hand poses, severe self-occlusions and self-similarities between fingers in the depth image.

A) Tracking based hand pose estimation

We focus our analysis on single frame methods. However, for completeness, we introduce Oikonomidis *et al.* [106] which proposed a tracking approach and, therefore, need a ground-truth initialization.

They formulated the challenging problem of 3D tracking of hand articulations as an optimization problem that minimizes differences between hypothesized 3D hand model instances and an actual visual observations. Optimization was performed with a stochastic method called Particle Swarm Optimization (PSO) [60]. Figure 2.11 illustrates their pipeline, where they first extracted the region of interest of the hand from a depth image and then fitted a 3D hand model using PSO. For an image at step t, the model is initialized using the final one found from the image $t-1$.

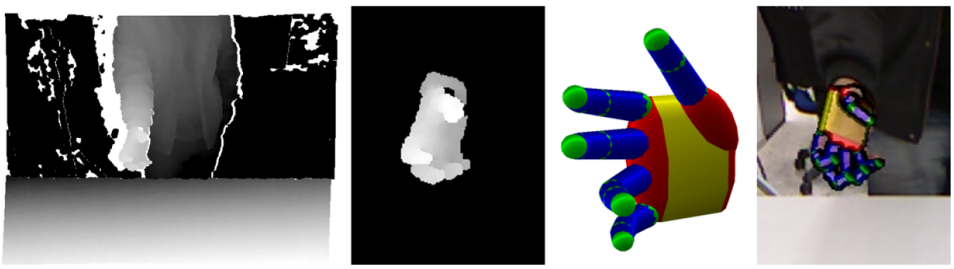


Figure 2.11 – *Pipeline illustration of Oikonomidis et al. [106]. (a) The current depth image. (b) First, the region of interest of the hand was extracted. (c) Second, the proposed method fitted the retrieved hand model from the last depth image (d) to the current one to recover the hand pose. Image reproduced from [106].*

Manual initialization may provide an unfair advantage but single frame methods are still competitive and in most cases even outperform tracking based approaches. One reason is that single frame methods re-

initialize themselves at each frame, while trackers cannot recover from continuous errors.

B) Single frame based hand pose estimation

Many recent approaches exploit the hierarchy of the tree structure of the hand model. Tang *et al.* [144] splited the hand into smaller sub-regions along the hand topological tree creating new latent joints. Using the Random Decision Forest algorithm, they perform coarse-to-fine localization of finger joints as depicted in Figure 2.12.

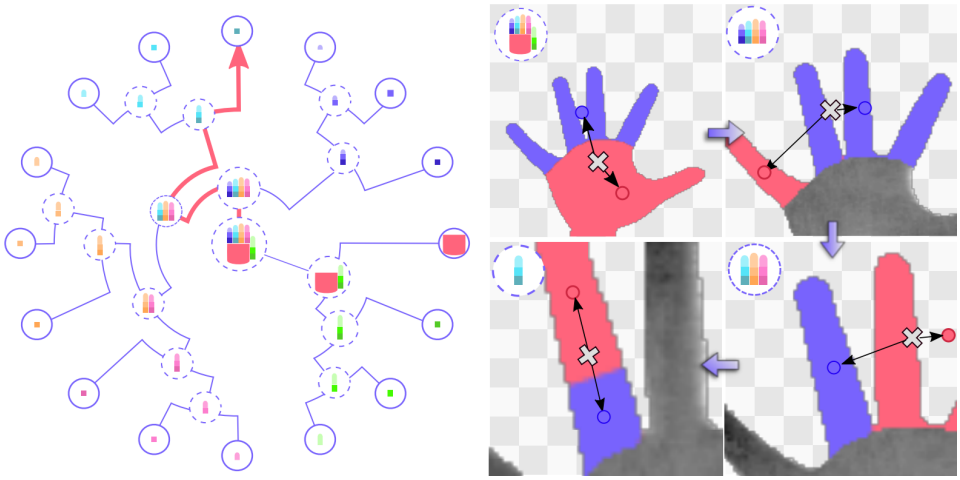


Figure 2.12 – *Tang et al. [144] can be viewed as a search process, guided by a binary tree model. Starting from the root of the tree, they minimize the offset to its children at each level until reaching a leaf node which corresponds to a hand skeletal joint position. For simplicity, the figure only show the searching process for one joint. Image reproduced from [144].*

Tang *et al.* [145] extended their idea using an energy function aiming to keep only the best partial poses through optimization iterations. Sun *et al.* [140] use a hierarchical hand pose regression from the palm to the fingertip locations. Yang *et al.* [166] proposed to use specialized hand pose regressors by, first, classify an incoming depth hand image based on a finite hand pose vocabulary and trained separate pose regressors for each classes.

All these approaches require multiple predictors, one for each joint, finger or yet hand pose classes and often multiple regressors for different steps of the algorithm. Thus the number of regression models ranges from 10 to more than 50 different models that have to be trained and evaluated.

Deep neural networks allowed overall improvement in many Computer Vision tasks. In 2015, Oberwerger *et al.* [101] evaluated several Convolutional Neural Network (CNN) architectures to predict 3D joint locations of a given a hand depth map. They stated that a constrained prior on the 3D pose can be introduced in the form of a bottleneck layer after the CNN (see in Figure 2.13). This method has significantly improved the accuracy and the reliability of the predictions.

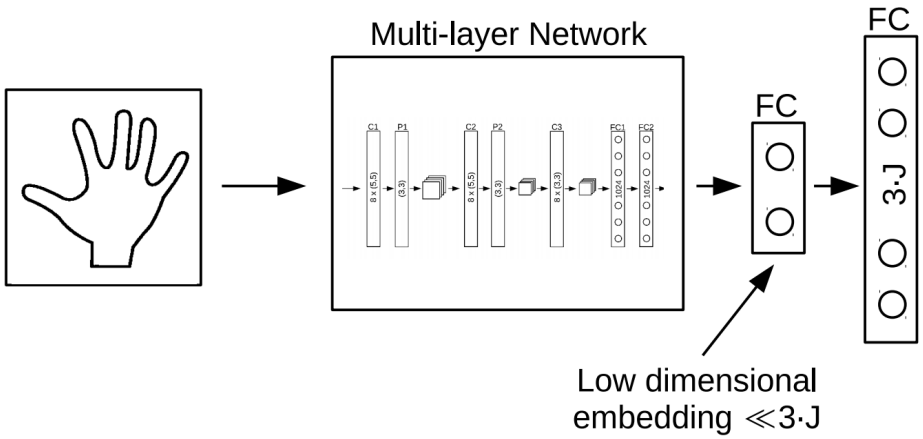


Figure 2.13 – *Oberwerger et al. [101] evaluated the use of a low dimensional embedding layer with less neurons than the output layer to incorporate a constrained pose prior. Image reproduced from [101].*

Zhou *et al.* [177] went further by integrating the real physical constraints into a CNN and introducing an additional layer that penalized unnatural predicted poses. Those constraints had to be defined manually.

Beside, several works integrated the hand model hierarchy into a single CNN structure. Ye *et al.* [170] introduced a spatial attention mechanism based on Convolutional Neural Network (CNN) that specializes on each joint and an additional optimization step to enforce kinematic constraints. Guo *et al.* [44] trained a set of networks for different spatial regions of the image and Madadi *et al.* [86] used a tree-shaped CNN architecture where each branch focus on one finger. Neverova *et al.* [97] combined a hand part segmentation based on CNN followed by a regression to perform joint locations. Unfortunately the segmentation has shown being sensitive to sensor noises.

Several representation of the input depth image has also been investigated. Deng *et al.* [26] converted the depth image to a 3D voxel vol-

ume and used a 3DCNN to predict joint locations. However, the 3DCNN showed a low computational efficiency. Beside, instead of directly predicts the 3D joint locations, Ge *et al.* [40] used multiple CNNs to predict heatmaps from different projections of the depth image and trained distinct CNNs for each projection as depicted in Figure 2.14.. This approach required a complex post-processing step to reconstruct a hand pose model from the heatmaps.

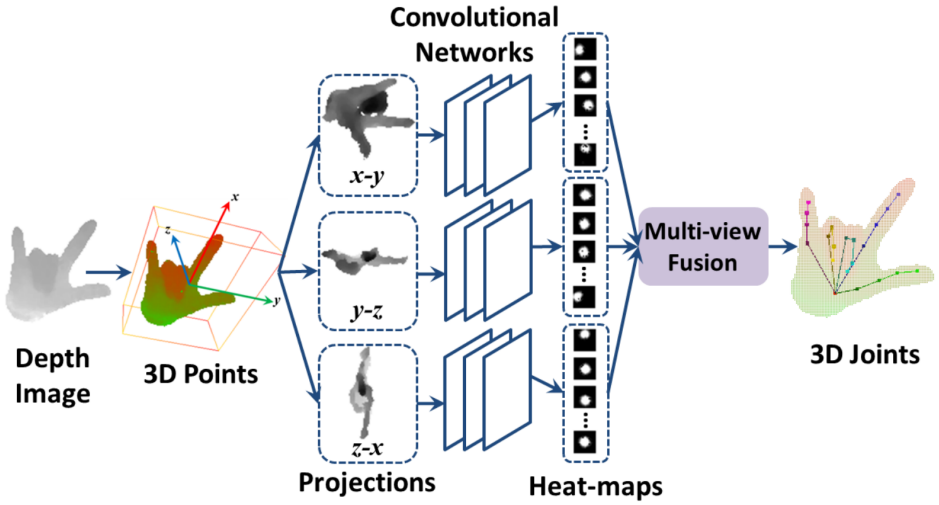


Figure 2.14 – *Overview of Ge et al. [40]. They generated heat-maps for three views by projecting 3D points onto three planes. Three CNNs are trained in parallel to map each view image to a heatmap. Finally, heatmaps are fused to estimate 3D hand joint locations. Image reproduced from [40].*

2.5 Datasets for hand gesture recognition

In recent years, the field of hand gesture analysis from RGB-D sensors has grown quickly, as it can facilitate a wide range of applications in Human-Computer Interaction, grasping taxonomy or yet daily life activities analysis. Compared to action and activity analysis, gesture analysis often does not need to deal with the whole body but only focuses on the hand region. Similarly to hand pose estimation, hand gesture datasets are necessary for the reliable testing and comparison of hand gesture recognition algorithms. The last few years, several RGB-D hand gesture datasets have been made publicly available. Most of them are reviewed here.

As shown in Table 2.2, these datasets have been collected for different

purposes in distinct scenarios and contain several specific characteristics listed bellow:

- Context: hand gesture datasets can be divided into four main context of study: *Grasp* taxonomy introduced by Feix *et al.* [35], *Sign language* datasets, *Hand gesture* datasets, a sub-field of study is *In car scenario* hand gesture analysis, finally, *Daily life activities* datasets.
- Size: with the arrival of data hungry algorithms, the number of samples available in a dataset has become an important factor.
- Spatial focus: if generally, hand gesture analysis focus on the region of interest of the hand. Meanwhile, some datasets, generally with the purpose of sign language recognition, focus on the upper-body or even the whole body motions.
- Temporal: datasets can also be divided in two groups following if they contain only static gestures (one sample is one image) or dynamic gestures (one sample is a sequence of images). Studying dynamic gestures is harder as it needs to study the temporal aspect of gestures in addition to its spatial one.
- Sensors: In the field of hand gesture recognition, the choice of the depth camera is primordial. Since very recently, a large amount of datasets have been made using the Kinect 1. Unfortunately, its low resolution do not allow a precise capture of the hand shape. To overcome this statement, several researchers created new datasets using newer short range camera with better resolutions.
- Data: in addition to RGB images, recent datasets provide depth maps. Very recently, following statements made in the field of action recognition, few datasets provide hand skeletal data in the form of 3D points referring to hand joints.

Additionally, Table 2.2 provides the size of the dataset vocabulary. We note that more a dataset contains a large vocabulary of gestures, more the recognition process is challenging.

Table 2.2 – *Summary of the most popular hand gesture datasets.*

Dataset	Context	Size voc.	Spatial focus	Temporal	Depth sensors	Samples	Data	View	Year	Av.
Yale [14]	Grasp	33	Hand	Dynamic	RageCams	18,210	RGB	First	2015	✓
UT Grasp [15]	Grasp	17	Hand	Dynamic	GoPro	–	RGB	First	2015	✓
GUN-71 [121]	Grasp	71	Hand	Static	Intel Senz3D	12,000	RGB, Depth	First	2015	✓
ASL finger-spelling [114]	Sign language	24	Hand	Static	Kinect 1	48,000	RGB, Depth	Third	2011	✓
MSR Gesture 3D [66]	Sign language	12	Hand	Dynamic	Kinect 1	336	Depth	Third	2012	✓
UESTC-ASL [18]	Sign language	10	Hand	Static	–	1,100	RGB, Depth	Third	2013	✗
ChaLearn [33]	Sign language	20	Whole body	Dynamic	Kinect 1	7,754	RGB, Depth, Body pose	Third	2014	✓
Marin *et al.* [88]	Sign language	10	Hand	Static	Kinect 1, LMC	1,400	Depth, 3D fingertip positions	Third	2014	✓
HUST-ASL [36]	Sign language	34	Hand	Static	Kinect 1	5,440	RGB, Depth	Third	2017	✓
Gesture [78]	Sign language	14	Upper-body	Dynamic	–	126	RGB	Third	2009	✓
NTU Hand Digit [119]	Hand gesture	10	Hand	Static	Kinect 1	1000	RGB, Depth	Third	2011	✓
SKIG [81]	Hand gesture	10	Hand	Dynamic	Kinect 1	2,160	RGB, Depth	Third	2013	✓
Xu *et al.* [165]	Hand gesture	10	Hand	Dynamic	LMC	1,600	Hand pose	Third	2014	✗
Wang *et al.* [152]	Hand gesture	10	Hand	Static	Kinect 1	1000	RGB, Depth	Third	2015	✓
Handicraft [84]	Hand gesture	10	Hand	Dynamic	LMC	300	Hand skeleton	Third	2016	✓
Ohn-Bar *et al.* [105]	In car scenario	19	Hand	Dynamic	Kinect 1	886	RGB, Depth	Third	2014	✓
NVIDIA [94]	In car scenario	25	Hand	Dynamic	SoftKinetic	1,532	RGB, Depth	Third	2016	✓
ADL [110]	Daily activities	18	Hand	Dynamic	GoPro	10 hours	RGB	First	2012	✓
WCVS [92]	Daily activities	10	Hand	Dynamic	Kinect 1	–	RGB, Depth	First	2014	✓
Daily H-O Actions [38]	Daily activities	45	Hand	Dynamic	Intel RealSense	1,175	RGB, Depth, Hand skeleton	First	2017	✗

In the field of grasp taxonomy, RGB images is still used [14, 15] as 2D images are easily obtained from common devices. Beside, Rogez *et al.* [121] created a grasp dataset providing depth images in addition to RGB but they only provide static gestures.

Several static hand gesture datasets [114, 18, 88, 36, 119, 152] have been introduced since 2011. They all contain two drawbacks: static gestures do not allow to study the temporal aspect useful for new HCI applications and they all has been captured using the kinect 1 [61] which has a low resolution.

Beside, dynamic hand gesture datasets [66, 81, 105, 92] have also been introduced using the Kinect 1 in the past. As the hand is a small and complex object with many self-occlusions, high resolution provided by new short range devices is needed. In addition, the Chalearn dataset [33] study sign language recognition with visual cues from the whole body and, so, is not suitable for fine-grained hand gesture analysis.

The NVIDIA Hand gesture [94] dataset aims to study dynamic hand gesture recongition in car and provides depth map sequences from a high resolution short range device. The Handicraft dataset [84] has been created to investigate the use of fine-grained hand gestures using hand skeletal data sequences. The dataset created by Xu *et al.* [165] contains also hand pose sequences but is not publicly available. While, the Daily Hand-Object Actions dataset [38] has been introduced in 2017 and is not yet publicly available.

2.6 Related works on hand gesture recognition

Due to the high number of potential applications and the increasing amount of publicly available datasets, many works have been proposed in the literature to address the problem of hand gesture recognition.

In this section, we first focus our review on the pre-processing steps on depth images which generally requires the extraction of the region of interest of the hand. Second, we introduce spatial features used in the literature to perform static and dynamic hand gesture recognition. Third, we review the proposed methods which consider the temporal aspect

of gestures, then, the main classification algorithms used for hand gesture recognition are listed. Finally, we introduce new advances in gesture recognition using deep learning approaches.

2.6.1 Pre-processing steps for hand localization

In the field of hand gesture recognition, pre-processing steps include a set of techniques and methods used in order to localize and segment the region of interest of the hand from the background, while trying to reduce noises from raw data. The quality of data pre-processing presents a challenge and it can significantly affect the performance of an algorithm. Supancic *et al.* [141] have shown that hand segmentation can be very challenging depending on the scenario, as from a complex background, and that an inaccurate result on this step causes poor gesture recognition.

Generally, hand localization techniques based on RGB images use skin color detection. They often give a fair result in simple contexts as skin tones are typically distinct from background colors. However, RGB based methods remain highly sensitive to illumination, individual differences and backgrounds. We refer to Schmugge *et al.* [127] for an overview of skin detection methods based on RGB images.

The arrival of depth sensors bringing another dimension has made possible to overcome certain challenges. Hand segmentation based on depth map can be done by thresholding the depth map [143, 142] – called Otsu's method – and region growing or contracting [66]; sometimes followed by an iterative refinement step [63] as depicted in Figure 2.15. In these cases, the hand is typically assumed to be the closest object to the camera.

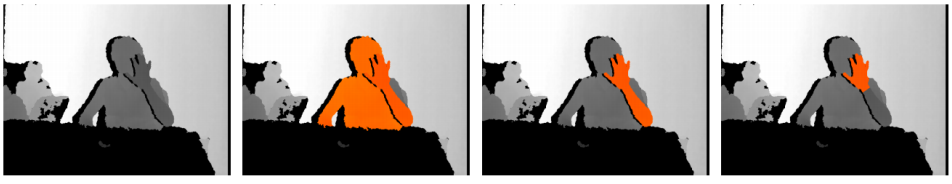


Figure 2.15 – *Otsu's method steps for hand segmentation from a depth image. (a) initial depth image. (b) segmented human body. (c) initial hand region. (d) refined hand region. Image reproduced from [66].*

Oberweger *et al.* [101] extended this idea by refining the extracted region using a cube around the center of mass and rejecting outsider points. However, this method can involve holes in the resulting map.

Inspired by Shotton *et al.* [134], Tompson *et al.* [147] trained, as a pre-processing step, a Random Decision Forest (RDF) to perform binary classification of each pixel of a depth map as either belonging to the hand or the background. If having an accurate hand mask is crucial for the task of hand pose estimation, their method seems over-engineered for hand gesture recognition.

2.6.2 Spatial features extraction

In this section, we introduce several descriptors proposed in the literature, which are computed from RGB or depth images to extract relevant information allowing to perform hand gesture recognition. Those descriptors can be divided in three groups: grid-shaped data descriptors which are used in several fields of computer vision, hand shape based descriptors which are specifically created for hand gesture analysis and, finally, skeletal features computed using outputs from a hand pose estimation system.

A) Grid-shaped data descriptors

Filters and gradients based descriptors have been widely used in many computer vision tasks. Van den Bergh *et al.* [150] used the Average Neighborhood Margin Maximization method [153] which computes Haarlet coefficients on single image. They combined features extracted from RGB and depth images but showed no improvements in adding the depth information. Pugeault *et al.* [114] used both grey-scale and depth images of a hand convolved with Gabor filters at different scales for the recognition of static hand gestures. Well known grid-shaped data descriptors, such as the Histogram of oriented Gradients (HoG) [21], the Scale-Invariant Feature Transform (SIFT) [83] or the Speeded Up Robust Features (SURF) [8] have been also used for the recognition of hand gestures in the past. SURF features have been directly exploited by Yao et Li [169] and Bao *et al.* [5] for, respectively, a static and a dynamic hand gesture recognition based

on RGB images. While the robust local feature descriptor SURF used histograms of gadrients on previously detected key points, Takimoto *et al.* [143] computed the SURF features after a hand extraction step from depth images, by replacing the key point detection step. Next, they divided the region of interest of the hand into 8×8 blocks of pixels, and computed histogram of gradients for each pixel block. Finally, Principal Component Analysis (PCA) was applied for feature dimensionality reduction, resulting in a new descriptor robust to scale and rotation of the hand. Dardas *et al.* [22] proposed a Bag Od Word (BoW) technique, which uses the SIFT features extracted from grey-scale images in addition to a vector quantification which maps keypoints into a unified dimensional histogram vector after K-means clustering. Their method is depicted in Figure 2.16.

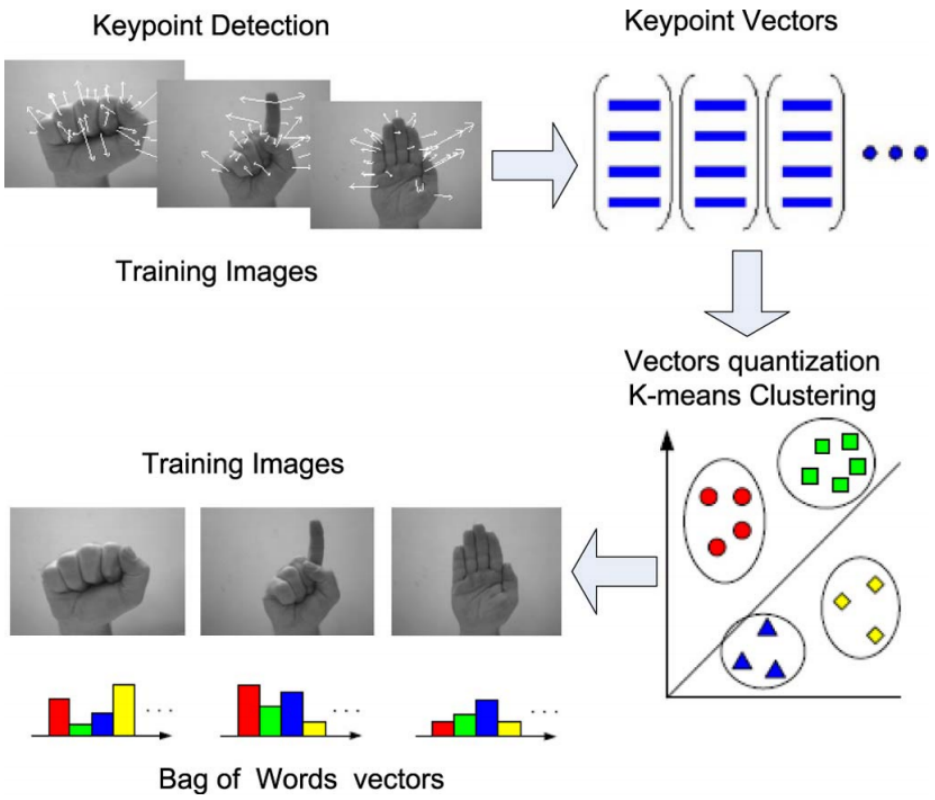


Figure 2.16 – *Overview of the method from Dardas et al. [22] SIFT features are extracted from grey-scale images and used as words to create a BoW dictionary. Image reproduced from [22].*

Zhang *et al.* [175] proposed a depth-based descriptor inspired by the HoG feature, called Histogram of 3D facets, and used it to perform recognition of static hand gestures.

Another group of depth-based methods exploits an occupancy pattern, like Kurakin *et al.* [66] who proposed to used cell and silhouette features extracted from thresholded depth maps of hands depicted in Figure 2.17. They split the image into square cells of equivalent sizes. For each cell, they computed the occupied area and the average depth, which they called a cell occupancy feature. For the silhouette feature, they divided the image into multiple fan-like sectors and computed the average distance of each contour segment from the center. The dimensionality of the obtained feature vector is then reduced with PCA.

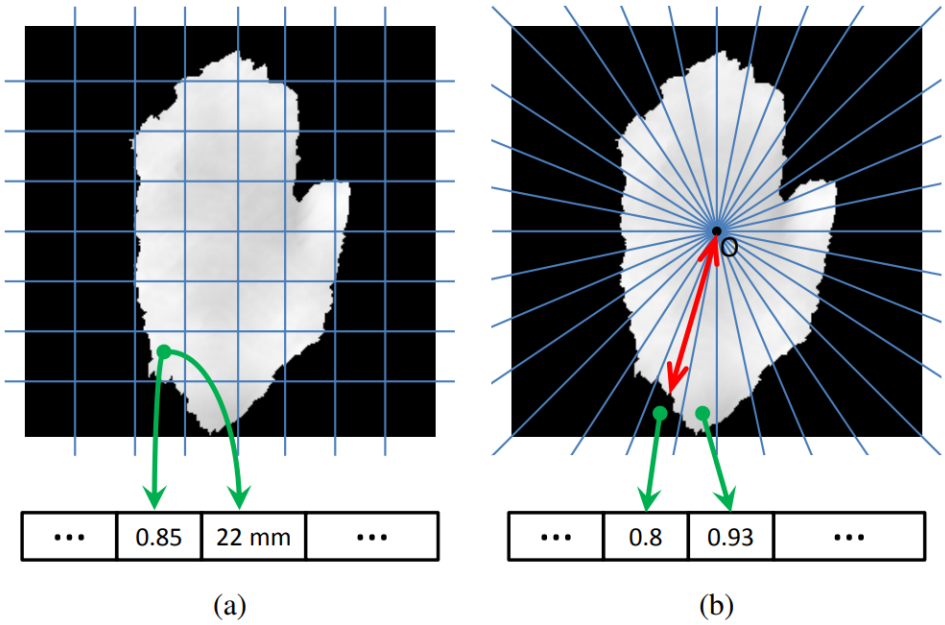


Figure 2.17 – *Overview of the method from Kurakin et al. [66]. Occupancy pattern based feature extraction. (a) They computed the occupied area of each cell as well as the average depth for the non-empty cells. (b) Using the silhouette, they divided the entire image into fan-like sectors. For each sector, they computes the average distance from the part of the contour inside this sector to the origin center of the mass of the region of interest of the hand. Image reproduced from [66].*

B) Hand shape descriptors

Beside general image descriptors, researchers have been actively looking for strong and robust descriptors designed specifically to describe the global hand shape and to perform hand gesture recognition.

Binary segmentation masks – called silhouettes – are often used for recognition and matching in static static hand gestures related problems. In the simplest case, they are obtained from color images or depth maps

as a result of background subtraction. Different shape descriptors can be further extracted from the extracted silhouettes.

Ren *et al.* [119] represented hand silhouettes as time-series contour curve, as depicted in Figure 2.18. In their framework, curves are compared using a Finger Earth Mover's Distance, originally proposed for measuring distances between signatures of histograms. They used the depth image to improve the hand segmentation step.

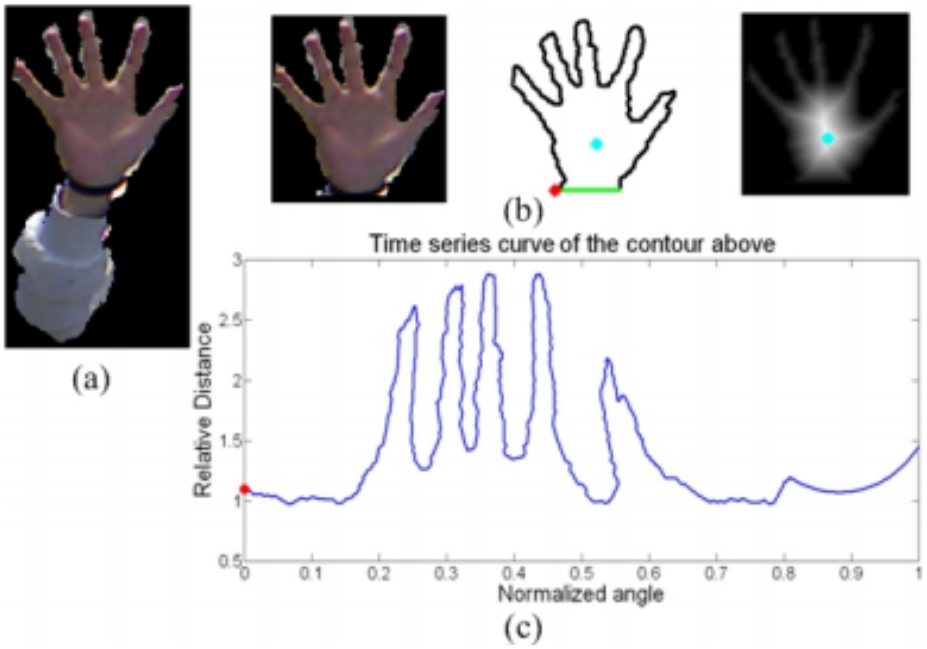


Figure 2.18 – *Overview of the method from Ren et al. [119]. (a) Coarse hand segmentation by depth thresholding; (b) A more accurate hand detection with black belt; (b - c) A time-series curve representation is computed using an initial point (red dot) and the palm point (cyan dot). Image reproduced from [119].*

The time-series curve representation has been also used by Cheng *et al.* [18] to generate sets of curves representing each static gestures from the dataset vocabulary. Conseil *et al.* [20] used Hu moments to encode hand silhouette shapes with invariance to translation, scale and rotation and perform a static hand gesture recognition.

At the early stage of hand gesture recognition, Shimada *et al.* [133] extracted a hand silhouette from a RBG image and compared it to predefined 3D models to infer the hand pose and use it to recognize static hand gestures. Later, Sudderth *et al.* [138] used edges and a hand silhouette in addition to a predefined hand model to extract the 2D hand topology, i.e. fingers and palm locations.

Stergiopoulou and Papamarkos [137] used a Self-Growing and Self-Organized Neural Gas (SGONG) network to identify the hand palm and the fingers, based on binarized hand image extracted from RGB images. Their method is depicted in Figure 2.19. They computed higher level 2D features from the hand topology such as angles, palm position and number of fingers raised to perform the recognition of static hand gestures. Instead of using a SGONG network, Wang *et al.* [152] studied the hand

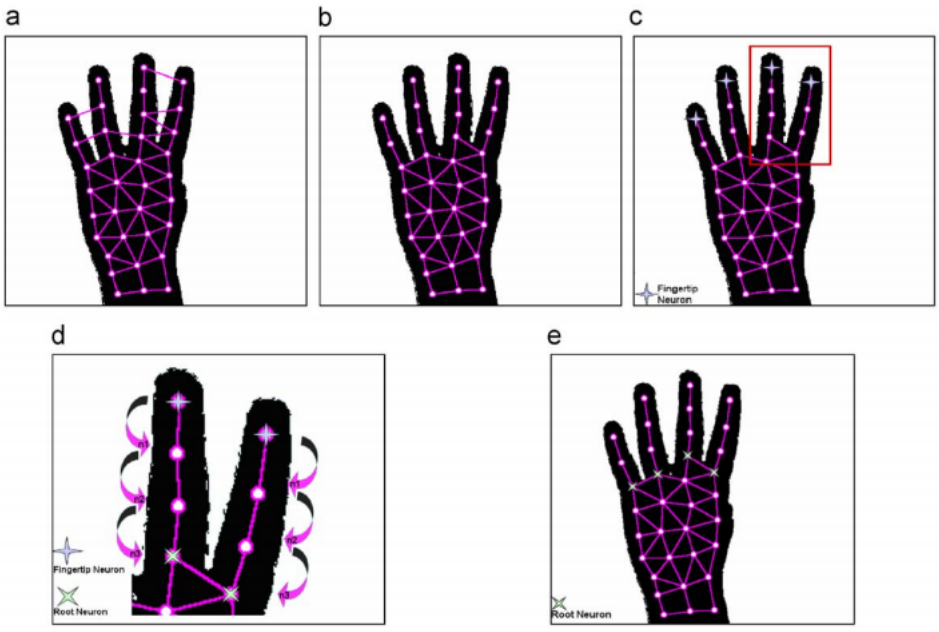


Figure 2.19 – *Overview of the method from Stergiopoulou and Papamarkos [137]. They use a SGONG network to identify the hand palm and the finger tips. (a) first, a grid of neurons is computed using a SGONG. (b) second, connection that go through the background are removed. (c) third, they determine the fingertip neurons. (d - e) finally, they successively determine the finger neurons and the root neurons. Image reproduced from [137].*

shape classification using a superpixel representation [2] depicted in Figure 2.20. They also extended the Finger Earth Mover's Distance [119] in order to use it on their representation and called it Superpixel Earth Mover's Distance.

With the arrival of inexpensive depth sensors, researchers started to used 3D information provided by such devices in addition to RGB images in order to extract precise 3D hand topology features. Inspired by Stergiopoulou *et al.* [137], Mateo *et al.* [89] extended their work using RGB to detect the hand and depth information in order to locate finger-

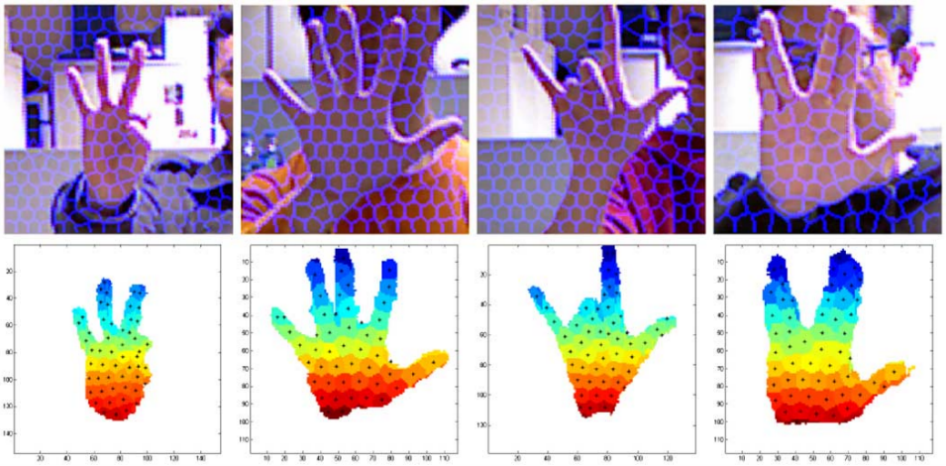


Figure 2.20 – *Overview of the method from Wang et al. [152]. Hand shape representation using superpixels. First row: superpixels on color textures. Second row: corresponding shapes represented by superpixels. Black dots indicate the centers of superpixels. Image reproduced from [152].*

tips using a k-d tree. Later, Dominio *et al.* [28] combined several types of hand anatomically meaningful descriptors computed on depth data, such as distances between estimated fingertips with a palm center or yet from a plane fitted on the palm, as well as curvature of the hand contour and the shape of the palm region. In order to find hand part locations, they employed skin-based segmentation.

Dong *et al.* [29] performed an analysis of static hand gestures for sign language recognition going in depth into the hand representation. They proposed a hierarchical mode-seeking method to locate positions of hand joints under kinematic constraints, segmenting the hand region into 11 natural parts: one for the palm and two for each finger.

C) Skeletal features

Motivated by the effectiveness of shape descriptors designed specifically for hand gesture analysis, recent works have shown their interest to extract skeletal features – called hand pose – from hand depth data. A review of hand skeletal feature extraction has been presented in Section 2.4. With advances in the field of hand pose estimation, skeletal features can be used instead of raw depth data for precise hand gesture recognition.

Beside, in the field of action recognition, Shotton *et al.* [134] proposed a real-time method to accurately predict the 3D positions of 20 body joints,

together called body skeleton, from depth images. Recently, several descriptors in the literature proved how the position, motion, and orientation of joints could be excellent descriptors for human actions [151, 155, 27].

In the field of hand gesture recognition, the use of hand skeletal data is at its beginning. Potter *et al.* [111] presented an early exploration of the suitability of using such hand skeletal data captured from a Leap Motion Controller (LMC) in order to recognize and classify precise hand gestures of the Australian Sign Language. They observed that the LMC has difficulty maintaining accuracy and fidelity of detection when the hands do not have direct line of sight with the controller.

Marin *et al.* [88] mixed depth features (i.e. multi-scale curvatures and correlations [28]) and skeletal data descriptors (i.e. fingertips angles, distances and elevation), respectively using a Kinect 1 and a LMC. Lu *et al.* [84] computed more features on the same data as palm direction and normal, adjacent fingertip distances and angles to perform precise dynamic hand gesture recognition.

Very recently, Garcia *et al.* [38] studied the use of hand skeletal data for hand daily life activities analysis. They perform numerous experimentation over their new dataset showing better performance of their hand gesture recognition system (based on the JOULE method [54]) using hand skeletal data compared to depth data. Finally, an improvement has been shown while merging RGB, depth and pose data.

2.6.3 Temporal modeling

The dynamic hand gesture recognition task involves modeling the temporal aspect of gestures in addition to the feature extraction. The literature of dynamic hand gesture temporal modeling shows two distinct strategies:

- Creating descriptors which carry spatial *and* temporal information;
- Modeling sequences of spatial descriptors via temporal classifiers.

A) Spatio-temporal descriptors

Spatio-temporal descriptors are widely used to recognize gestures from videos. Several of these descriptors have evolved from their 2D ver-

sions, by augmenting existing descriptors with additional features extracted along the temporal dimension.

For example, Ohn *et al.* [104] introduced a 3D version of the HOG descriptor, called HOG^2, to handle videos and Klaser *et al.* [62] proposed similarly the HOG3D descriptor. Ohn *et al.* [105] experimented both descriptors to recognize dynamic hand gestures performed in a car scenario. They computed both descriptors on both depth and RGB videos, where depth data has showed better performances and merging both information increased their results.

Zhang *et al.* [174] introduced a spatio-temporal representation of hand depth sequences, called Edge Enhanced Depth Motion Map (E^{2}DMM). Their method is related to the Depth Motion Map (DMM) representation which has been used in the action recognition field [168]. According to Yang *et al.* [168], edge suppression is suitable for action recognition since the contours do not provide useful information to help distinguishing between different actions. However, both static hand pose and motions convey significant information in the perspective of dynamic hand gesture recognition. Following this statement, Zhang *et al.* [174] changed the edge suppression term to an edge enhancement term. The E^{2}DMM descriptor is depicted in Figure 2.21. Finally, they computed the HoG descriptor [21] on the E^{2}DMM. As characterizing a whole hand gesture sequence using a single representation can lead to miss-classification due to inverse gesture, they used a temporal pyramid. The principle of the temporal pyramid is to divide a sequence into n sub-sequences at each n^{th} level of the pyramid. The final representation is the concatenation of all spatio-temporal descriptors computed on all sub-sequences. This strategy allows to distinguish the beginning, the middle and the end of gestures.

B) Sequence modeling

Once a spatial descriptor has been computed in a frame-wise or after a sub-sequences-wise segmentation, the sequence is modeled by statistical algorithms that take a sequence as input. In the category of sequence modeling methods, Hidden Markov Models (HMMs), Hidden Conditional

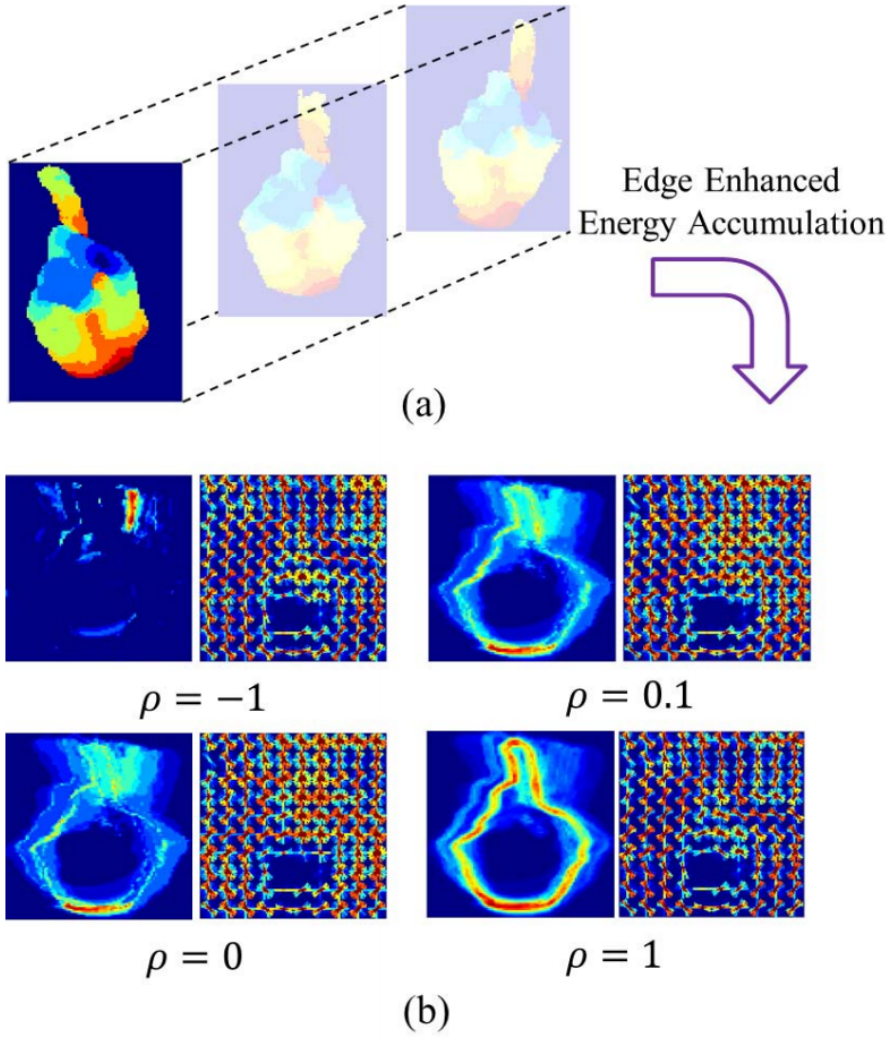


Figure 2.21 – *Overview of the method from Zhang et al. [174]. (a) A depth video showing a hand gesture from the American Sign Language. (b) Accumulative Motion Maps and corresponding visualizations of the HoG descriptor following different degree of edge enhancement. When $\rho = -1$, it degenerates to DMM and when $\rho > 0$, it is in the form of E^2DMM. Image reproduced from [174].*

Random Fields (HCRFs) and their extensions have proven to be efficient in many sequential recognition tasks. While HMM models joint probability of states and observed features, the parameters in HCRFs model are conditioned only on observations.

The HMM is a stochastic algorithm governed by a finite number of probabilistic states and a set of probabilistic functions associated with each state. The idea is to create a probabilistic model for each class of the vocabulary of the current classification problem. Given an input sequence S, each HMM outputs the probability that S belongs to their associated class which make HMM a temporal classifier and not only a modeler. In the context of dynamic hand gesture recognition, each state could represent a set of possible hand positions [17, 71]. The state transitions represent the probability that a certain hand position goes into another. An extension of HMM called action graph [73] has been used by Kurakin *et al.* [66]. Compared to HMM, an action graph has the advantage that it requires less training data and allows different actions to share the states. The HCRF algorithm is driven by the idea to eliminate unrealistic assumptions, that can build a HMM, and handling long term dependencies. Wang *et al.* [159] experimented the effectiveness of HCRF over HMM on the challenge of arm gesture classification. In addition, Manitsaris et al. [87] combined a HMM and a Dynamic Time Warping (DTW) approach permitting to early recognize and predict the gestures.

2.6.4 Classification

Classification algorithms are systems that learn to map an unknown input data to one label from a finite vocabulary. Supervised machine learning takes a set of labeled training data which is used to infer the mapping function. It exists a large variety of classifier and we refer to Kotsiantis *et al.* [64] for a review of classification techniques. In this section, we introduce three of the main used classification algorithms which are the Support vector machine (SVM) [49], the k-Nearest Neighbor (k-NN), and the Random Decision Forest (RDF) [12] algorithms.

SVM [49] finds the optimal hyperplane to separate the training data following their known label. An SVM maximizes the margin around the separating hyperplane. Optimization techniques are employed to find the optimal hyper plane. SVM is, so far, the most used classifier in human behavior analysis and a baseline for many datasets [22, 175, 82, 88, 174].

k-NN is a simple statistical method, where an input data will be assigned to the most common class among its k nearest neighbors in the training set. The neighbors are defined following a similarity measure often computed on features extracted from raw data. Gupta *et al.* [45] used k-NN based on HOG and SIFT descriptors to classify static gestures. However, several experimentation on comparing the accuracy of k-NN against SVM [49] have shown that the performance of k-NN is comparatively lower [146, 118, 6].

RDF [12] consists of a learned set of randomized classification trees. Each of them is trained with a random subset of the original training data. Each binary classification tree is built by recursively partitioning the input data at each node, so as to reduce the entropy of the class distribution. At each node, a random subset of features is selected and the threshold leading to the largest reduction in class distribution entropy is chosen. RDF [12] has been used by Pugeault *et al.* [114] for static hand gesture recognition based on Gabor filter response features.

2.6.5 Deep learning approaches

Like many research areas in pattern recognition, including the hand pose estimation field, deep learning approaches have shown particularity good performance for hand gesture recognition. Their ability to learn relevant spatial and/or temporal features in addition to play the role of classifier, has been studied last years.

Convolutional neural networks [69] designed to take images as input has been used for static hand gesture recognition using RGB data [77, 96] and/or depth maps [72].

Neverova *et al.* [98] designed a multi-modal deep learning framework which takes as inputs: RGB, depth, audio stream and body skeleton data (see in Figure 2.22). Their network captured several spatial information, such as motions of the upper body or the hand, at three distinct spatial scales in order to perform dynamic sign language recognition. Their framework classified each frame and the final label of a sequence was computed using a majority vote.

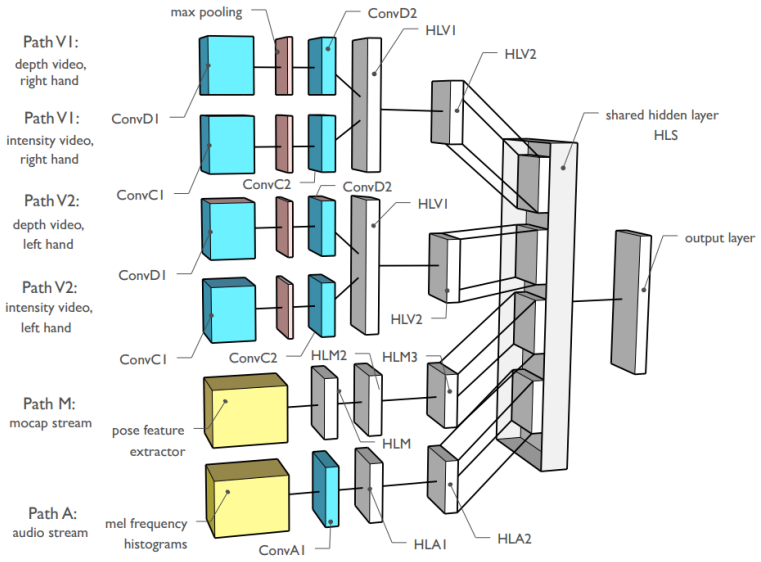


Figure 2.22 – *The single-scale multi-modal deep learning framework from Neverova et al. [98]. Individual classifiers are trained for each data modality (paths V1, V2, M, A). Image reproduced from [98].*

Molchanov *et al.* [93] proposed a dynamic hand gesture algorithm using a two-stream 3DCNN which takes as inputs stacked image gradients and depth maps to classify sequence of images. The 3DCNN is depicted in Figure 2.23.

They later enhanced their method and proposed a dynamic hand gesture algorithm – called R3DCNN [94] – using a larger 3DCNN, precedently defined by Karpathy *et al.* [59], to extract features from subsequences followed by a recurrent layer to model the temporal aspect of gestures. The 3DCNN was composed of eight convolutions with an increasing number of filters in order to get both spatial and temporal features on sequences of RGB and depth images. In addition, they used a Connectionist Temporal Classification [43] as the cost function. While it has been initially designed to perform prediction of sequence in an unseg-

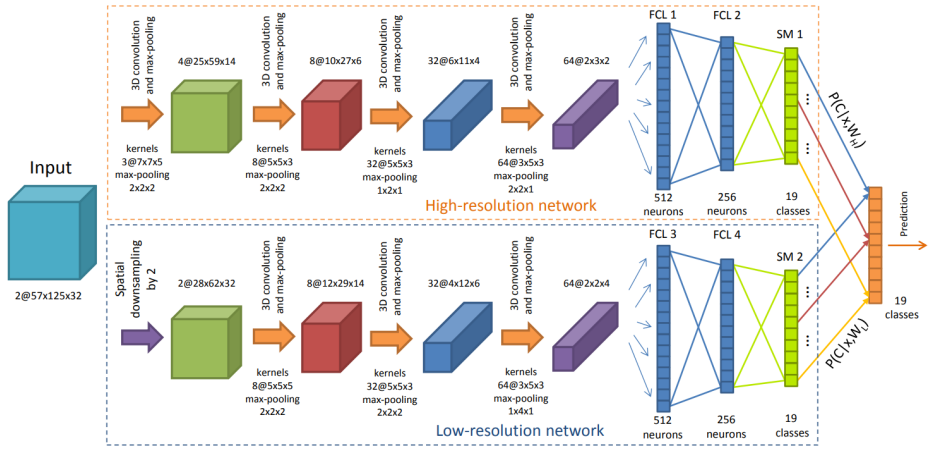


Figure 2.23 – *The 3DCNN architecture classifier from Molchanov et al. [93]. The inputs of the classifier were stacked image gradients and depth values. The classifier consisted of two sub-networks: a high-resolution network (HRN) and a low-resolution network (LRN). The two networks were fused by multiplying their respective class probabilities. Image reproduced from [93].*

mented input streams scenario, the CTC is applied here to perform online classification. Their framework is depicted in Figure 2.24.

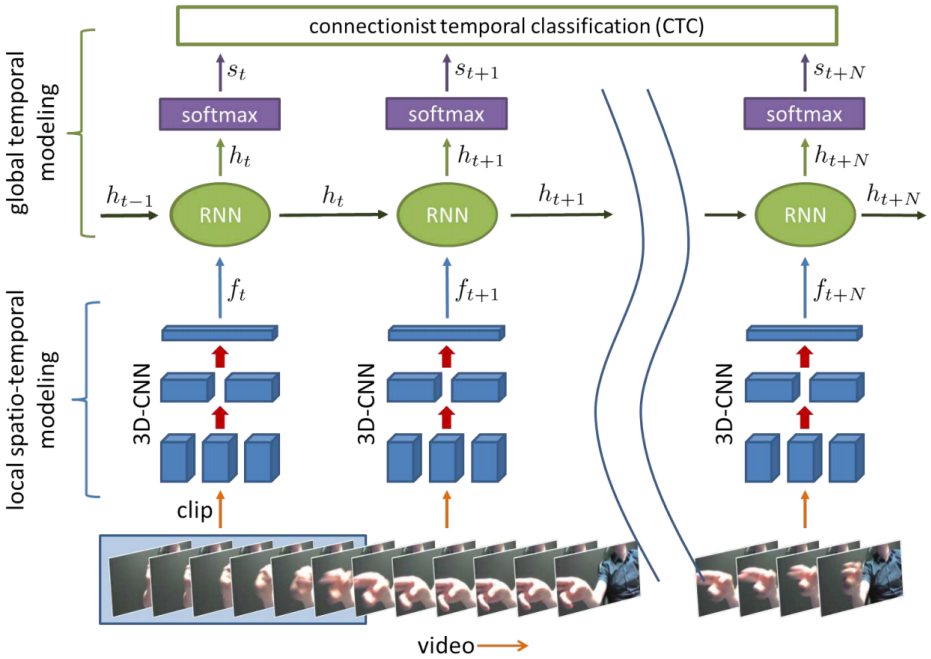


Figure 2.24 – *The hand gesture classification framework from Molchanov et al. [94]. A gesture video is presented in the form of sequence of short clips to a 3DCNN for extracting local spatial-temporal features. These features are input to a recurrent network. The recurrent network has a hidden state h_{t-1}, which is computed from the previous clips. The updated hidden state for the current clip, h_t, is input into a softmax layer to estimate class-conditional probabilities. During training, a Connectionist Temporal Classification is used as the cost function. Image reproduced from [94].*

To overcome the hungryness of deep learning algorithms, they pre-trained their model on the large-scale Sport-1M [59] human action recognition dataset. If they claimed to obtain real-time results, they used a powerful hardware configuration not suitable for public use.

The recognition algorithms of dynamic hand gestures based on skeletal data are not yet well represented in the literature. This is due to the fact that hand pose estimation methods begin only recently to be robust and efficient in challenging contexts. Lu *et al.* [84] used a neural based variant of the HCRF which were fed with features computed on hand skeletal data captured via a LMC.

In the meantime, the problem of modeling skeletal data sequences with deep neural networks has been studied in the field of action recognition. We introduce recent advances in skeletal sequence modeling using recurrent networks for completeness.

Wang *et al.* [156] used a two-stream Recurrent Neural Network (RNN) architecture for skeleton based action recognition. One stream was used in order to model temporal information while the other focus on spatial cues.

Garcia *et al.* [38] used a two-stacked Long-Term Short Memory (LSTM) network as a baseline for their hand action dataset. LSTM has shown better performance over all previous traditional methods. Du *et al.* [31] proposed to divide the human body skeleton in five meaningful parts and fed each one into a distinct RNN network. They used a bidirectionnal variant [128] of the LSTM in order to use past frames but also future one to model each time step of a sequence. The recurrent layers are then fused step by step to be inputs of higher layers. Their network is depicted in Figure 2.25.

Very recently, Liu *et al.* [80] defined a global context-aware attention LSTM networks for skeleton-based action recognition. The idea behind their approach is that an upstream recurrent network process an incoming sequence and update a context memory cell which allowed to extract the potential importance of each body joints in the sequence. A second LSTM

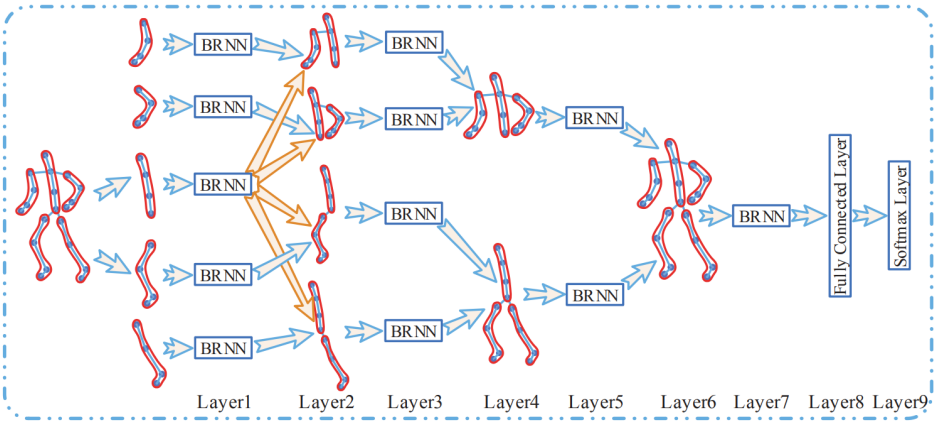


Figure 2.25 – *An illustration of the hierarchical recurrent neural network proposed by Du et al. [31]. The body skeleton is divided into five parts, which are fed into five bidirectional recurrent neural networks (BRNNs). As the number of layers increases, the representations extracted by the subnets are hierarchically fused to be the inputs of higher layers. A fully connected layer and a softmax layer are performed on the final representation to classify actions. Image reproduced from [31].*

performed the classification paying attention more precisely at some joints using the context memory.

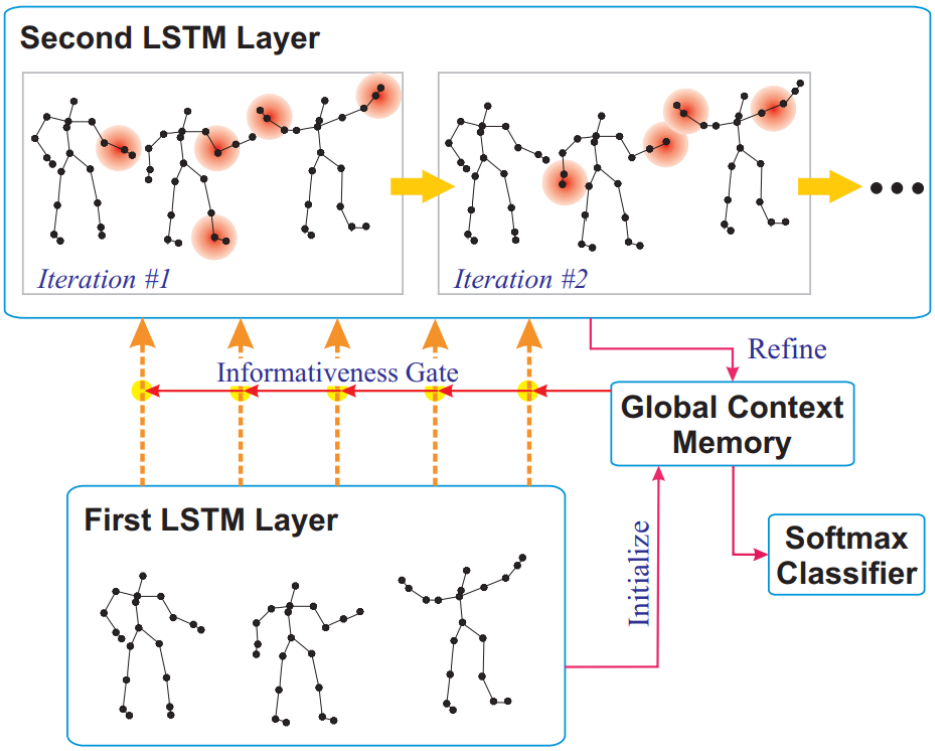


Figure 2.26 – *An illustration of the global context-aware attention LSTM network proposed by Liu et al. [80]. The first LSTM layer encodes the skeleton sequence and update the global context memory. The second layer performs attention over the inputs with the assistance of the context memory and generates an attention representation for the sequence. Image reproduced from [80].*

2.7 Discussion and conclusion

The 3D information given by depth sensors encouraged researchers to investigate depth maps for both tasks of hand pose estimation and hand gesture recognition. Modeling motions and deformation of the hand shape is considered to be more challenging that other human parts due to its smaller size, greater complexity and higher amount of self occlusions. Consequently, new datasets and new methods have been conducted last years taking advantages of short range depth camera which extract more accurate details in comparison with long range camera.

Hand pose estimation aims to retrieve the transformation of a predefined structured model of the hand from visual cues which represent the real hand shape of the user captured by a camera. In the field of action recognition, such human body skeleton improved considerably the performance of systems which aim to understand human behavior. Currently, the challenge of using hand skeletal data for hand gesture analysis is at its beginning. Researchers have been able to exploit skeletal data in the field of human behavior understanding since many years due to the maturity of the body pose estimation. On the other side, hand pose estimation is still very challenging due to the intrinsic characteristics of the hand.

To overcome challenges of hand pose estimation, current state-of-the-art methods use deep learning approaches. Briefly, deep learning is a family of learned based features algorithms based on neural networks. Very recently, using this approach allowed researchers to make a jump in robustness and efficiency in hand pose estimation. However, deep learning algorithms are data hungry and annotating hand pose dataset is very time consuming. Thus, only small amount of samples are available in publicly available datasets. In the end of 2017, Yuan *et al.* [172] will make available a hand pose dataset with more than two millions of samples hoping it will allow researchers to overcome some of the challenges of hand pose estimation.

In the field of hand gesture recognition, early proposed methods were using RGB data. However, extracting the region of interest of the hand

was very challenging, even with powerful skin detector algorithms, due to the highly sensitivity of algorithms to illumination, individual differences and backgrounds. In addition, the hand suffers from severe self-occlusions and RGB data allowed researchers to study the hand shape only in two dimensions. Similarly to the field of hand pose estimation, as soon as cheap depth sensors were available, researchers investigated ways to take advantage from the 3D information they provide. Proposed methods can be sorted from algorithms using general descriptors – which were already used to tackle others computer vision challenges – to hand specific feature extraction such as time-series curves describing the states of fingers and very recently the use of sequence of hand skeletal data for dynamic hand gesture recognition.

Similarly to hand pose estimation, methods using deep learning for the task of hand gesture recognition showed a jump in the robustness and the efficiency of new algorithms based on learned features compared to traditional handcrafted descriptors. However, in the same way, current available hand gesture datasets are small in size and strategies have to be used to overcome the hungriness of deep learning algorithms.

All these considerations motivate us to address the problem of hand gesture recognition according to two categories of methods: hand-crafted and deep learning approaches, while investigating all challenges raised in the literature.

Hence, in the following chapters, we first investigate precise and dynamic hand gestures using handcrafted descriptors on hand skeletal data. We also discuss the creation of our own challenging dataset. This dataset allowing us to study the *intra-class* and *inter-class* high similarities between hand gestures not yet invested in the literature.

In a second time, we study the evolution of the traditional recognition processes compared to deep learning approaches. We also detail technical elements of deep neural networks useful for hand gesture recognition.

Finally, we extend the analysis of dynamic hand gesture recognition mixing two deep learning approaches for, both, hand pose estimation and hand gesture recognition. We aim to take over the whole pipeline of the

process in order to propose an online hand gesture recognition system for real time applications.

3 Heterogeneous hand gesture recognition using hand-crafted descriptors on 3D skeletal features

Contents

3.1 Introduction

Using hand gestures as a Human-Computer Interaction (HCI) modality introduces intuitive and easy-to-use interfaces for a wide range of applications in virtual and augmented reality systems, offering support for the hearing-impaired and providing solutions for all environments using touchless interfaces. However, the hand is an object with a complex topology and has endless possibilities to perform the same gesture. For example, Feix *et al.* [35] summarize the grasping taxonomy and found 17 different hand shapes to perform a grasp. Grasping is a hand gesture where we need precise information about the hand shape if we want to recognize it. Other gestures, such as *swipes*, which are defined more by hand motions than its shape, are already commonly used in tactile HCI. Thus, the heterogeneity between useful gestures have to be taken into account in a hand gesture recognition algorithm.

To date, most reliable tools used to capture 3D hand gestures are motion capture devices, which have sensors attached to a glove delivering real-time measurements of the hand. However, they present several drawbacks in terms of the naturalness of the hand gesture and cost, in addition to their complex calibration setup process. Recently, effective and inexpensive depth sensors, like the Microsoft Kinect, have been increasingly used in the domain of computer vision. By adding a third dimension into the game, depth images offer new opportunities to many research fields, one of which is the hand gesture recognition. In recent years, many researchers [67, 154, 119, 18, 152, 114, 66, 175, 95, 33, 105, 93, 94] has studied 3D hand gesture recognition challenges using depth images.

Beside, in the field of action recognition, Shotton *et al.* [134] proposed a real-time method to accurately predict the 3D positions of 20 body joints, together called *body skeleton*, from depth images. Hence, several descriptors in the literature proved how positions, motions, and orientations of joints could be excellent descriptors for human actions. Following this statement, hand skeletal data could also handle precise information of the hands that HCI needs in order to use them as a manipulation tool.

Very recently, new devices, such as the Intel RealSense or the Leap

Motion Controller (LMC), provide, in addition to depth images, precise **skeletal data** of the hand and fingers in the form of a full 3D skeleton corresponding to 22 joints in $\mathbb{R}^3$ labeled as depicted in Figure 3.1. Potter *et al.* [111] presented an early exploration of the suitability of using such data from a LMC in order to recognize and classify precise hand gestures in Australian Sign Language. However, hand pose estimation from depth

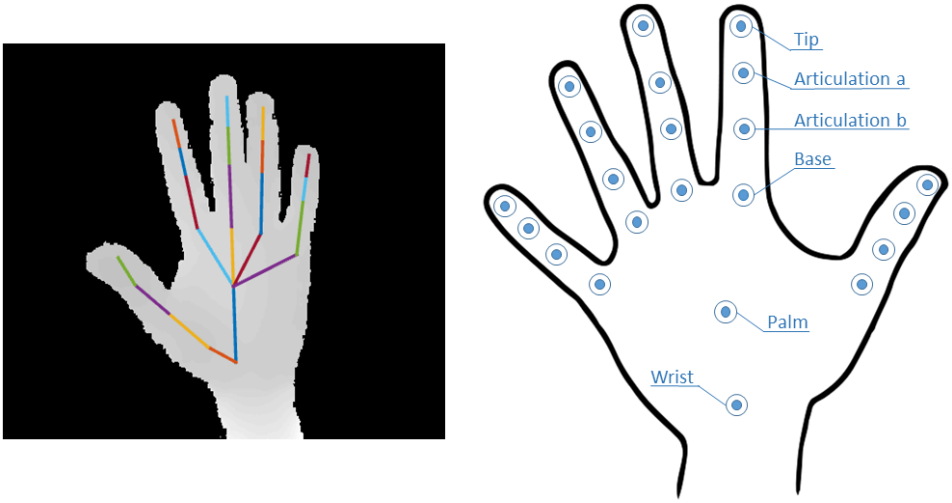


Figure 3.1 – *Depth and hand skeletal data returned by the Intel RealSense camera. The 22 joints include: one for the center of the palm, one for the position of the wrist and four joints for each finger represent the tip, the two articulations and the base. All joints are represented in* $\mathbb{R}^3$*. The Leap Motion Controller provides a similar hand skeleton.*

images remains a prominent field of research. Many issues still have to be solved: properly recognizing the skeleton when the hand is either closed, perpendicular to the camera, or without an accurate initialization, or when the user performs a quick gesture. The hand contains more joints than there are in the rest of the human body model of Shotton *et al.* [134] and is a smaller object. The hand has also a more complex structure. If an arm, a head or a leg have different shapes, the hand is composed of a palm and five similar fingers making its pose estimation more challenging.

In this chapter, we aim to study the use of hand skeletal data to perform dynamic hand gesture recognition from a set of heterogenous gestures. The idea behind the term "heterogenous" is that there are various gesture types. *Coarse* gestures that are defined by hand motions (e.g. *Swipes*), *Fine* gestures that are defined by hand shape variations (e.g. *Open the hand*) or even both. In addition, gestures can be either *Static* (e.g. *Show*

one finger) or *Dynamic* (e.g. *Grab something*). To fully understand and design a robust algorithm able to classify an unknown incoming gesture, motions and precise shape information about the hand have to be extracted.

3.1.1 Challenges

The development of a precise dynamic hand gesture recognition system, able to take into account the heterogeneity of possible gesture types, presents some important challenges.

First, the main challenge is the *intraclass* gesture dissimilarities. They come from *ad-hoc*, cultural and/or individual factors in the style, position and speed of gestures. Indeed, even after explanations and examples, two different subjects rarely perform the same gesture in the same way. These variations are caused by differences of dexterity, size or yet again culture. In fact, even a particular subject never perform the same gesture twice. The first obvious reason is that its position relative to the camera can change. Moreover, as a user performs a particular type of gesture multiple times, he makes it his own. It follows sometimes large differences with the example given at the beginning. An efficient algorithm should be able to recognize hand gestures independently of this factors.

In addition to *intraclass* gesture dissimilarities, an important factor in precise hand gesture recognition task is *interclass* similarities. They come from high similarities between different types of gestures. Furthermore, these similarities are exacerbated by deformations due to *intraclass* variations.

Finally, some hand gestures can only be described by hand shape variations through time. However, a hand is a small object with a high degree of freedom and with a high potential of self-occlusion. It is very hard to extract precise information of the hand shape based on data captured using old depth sensors with a low image resolution such as the first version of Microsft Kinect. In addition, the noise of depth images, self-occlusions and situational variations in the viewpoints make the study of hand shape very challenging. Nevertheless, new short ranged depth devices allow re-

searchers to get more precise hand capture (e.g. Intel RealSense or the SoftKinetic DS325).

3.1.2 Overview of the proposed method

To face challenges of dynamic hand gesture recognition, we introduce an original approach using three features computed on hand skeletal feature sequences to classify unknown hand gestures.

Our proposed method is a hand skeleton-based approach since we consider those features contain precise information about hand motions and shape variations information. In addition, new devices are able to directly provide us hand skeletal features. However, even if 3D joint positions of hand skeleton are available, the hand gesture recognition task is still challenging due to significant spatial and temporal variations in the way of performing a gesture.

First, we use a temporal pyramid to represent the dynamic aspect of gestures. We cut sequences in overlapping sub-sequences. On each sub-sequences, we compute three set of features: a set of direction vector which the hand is taking through the sequence, a set of rotation and a hand shape descriptor called *Shape of Connected Joints*. Those sets are then transformed into a statistical representation vector using a Fisher Kernel. The final gesture descriptor is the concatenation of the three statistical representation features computed for each sub-sequence.

Finally, a linear SVM is used to perform classification. Figure 3.2 summarizes the proposed approach.

3.1.3 Motivations

The main considerations that motivated our approach are:

- Precise hand gesture recognition is becoming a central key for different types of applications (e.g. virtual game control, sign language recognition, ...).
- Heterogeneous hand gesture recognition needs to fully understand hand shape variations and motions through gestures.

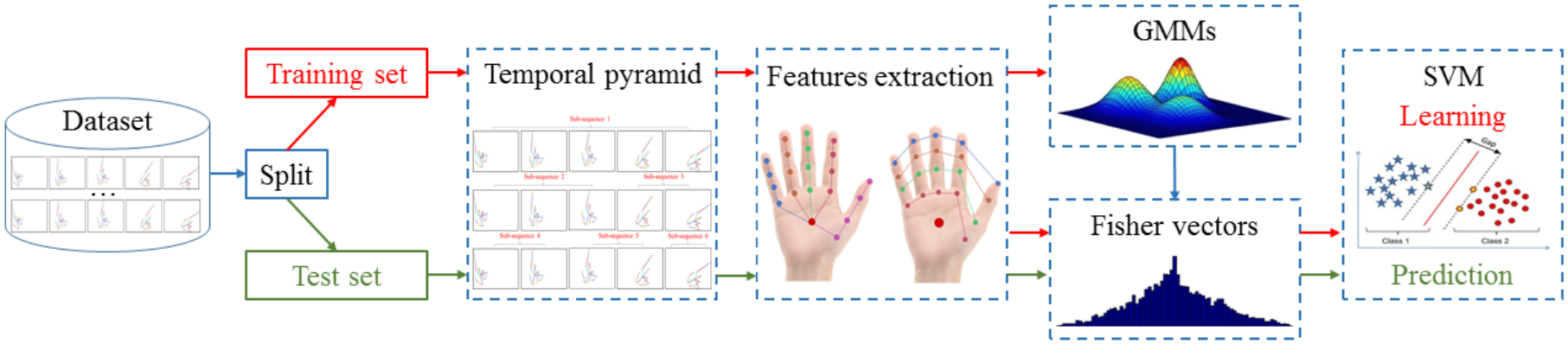


Figure 3.2 – *Overview of our hand gesture recognition approach using hand skeletal data. First, sequences are divided in overlapping sub-sequences to represent the dynamic aspect of gestures using a temporal pyramid strategy. On each sub-sequences, we computed three set of features: directions, rotations and a hand shape descriptor. A fisher kernel is use to transform those sets in a statistical representation. Finally, we use a SVM for hand gesture classification.*

- The use of skeletal features result in improvements in the field of action recognition.
- New short ranged depth devices and hand pose estimation methods allow to extract hand skeletal information.

3.2 The Dynamic Hand Gesture dataset (DHG-14/28)

Skeleton-based action recognition approaches have become popular as Shotton *et al.* [134] proposed a real-time method to accurately predict the 3D positions of body joints from depth images. Hence, several descriptors in the literature proved how the position, the motion, and the orientation of joints could be excellent descriptors for human actions. Collected datasets for action recognition purpose [158, 74, 130, 129, 74] provide the 3D body skeleton of the person performing the action in addition to depth images.

However, in the context of hand gesture recognition, there are no publicly released dynamic hand gestures dataset providing labeled sequences of depth *and* hand skeletal features.

In this section, we present the *Dynamic Hand Gesture 14 / 28* (DHG) dataset collected by De Smedt at al. [23], which provides hand skeletal sequences in addition to depth images. Such a dataset facilitate the analysis of hand gestures and open new scientific axes to consider[1].

3.2.1 Overview and protocol

The DHG-14/28 dataset contains 14 dynamic hand gesture types performed in two ways: using one finger or the whole hand (an example is shown in Figure 3.3). Each gesture is performed 5 times by 20 participants in 2 ways, resulting in 2800 sequences.

Sequences are labeled following their gesture, the number of fingers used and the performer. Each frame contains a depth image, coordinates of 22 joints – together called a *hand skeleton* – both in the 2D image and in the 3D camera space. The Intel RealSense short ranged depth camera is

[1] Available on: http://www-rech.telecom-lille.fr/DHGdataset

used to collect the dataset. Depth images and hand skeletons were captured at 30 frames per second. Depth images have a 640×480 resolution. The length of sample gestures ranges goes from 20 to 50 frames.

Fothergill *et al.* [37] investigated the problem of the most appropriate semiotic modalities of instructions for conveying to performers the movements the system developer needs to perform. They found out that a gesture recognition algorithm not only needs examples of desired gestures but also in order to cope with a wide array of users, the dataset must include common desired variants of the gestures. To achieve a good correctness in our dataset, we use two modalities to explain what we waited from our performers. First, the register explains in an abstractive way the gesture (e.g. to perform a swipe gesture with one finger: "You're going to mime a swipe in the air with only one finger"), then we were showing them a video of someone performing the gesture.

In terms of hand pose estimation, much attention has been received over the last two years in the computer vision community [144, 132]. The Software Development Kit (SDK) released for *Intel RealSense* F200 provides a full 3D skeleton of the hand corresponding to 22 joints labeled as depicted in Figure 3.1. However, we note that the sensor still has trouble to properly recognize the skeleton when the hand is closed, perpendicular to the camera, without a correct initialization or when the user performs a quick gesture. Our participants were asked to start each sequence by one or two seconds with the hand well opened in front of the camera. This may be necessary for some state-of-the-art hand pose estimation algorithms requiring a hand shape initialization step. For those who do not need initialisation frames, we manually labeled effective beginnings and ends of each gesture sequences.

3.2.2 Gesture types included

Gesture types included in the DHG dataset are listed in Table 3.1. Most of them have been chosen to be close to the state-of-the-art, like the VIVA challenge dataset [105]. Nevertheless, we removed the differentiation between normal and scroll swipes as you can find it in our number-of-fingers

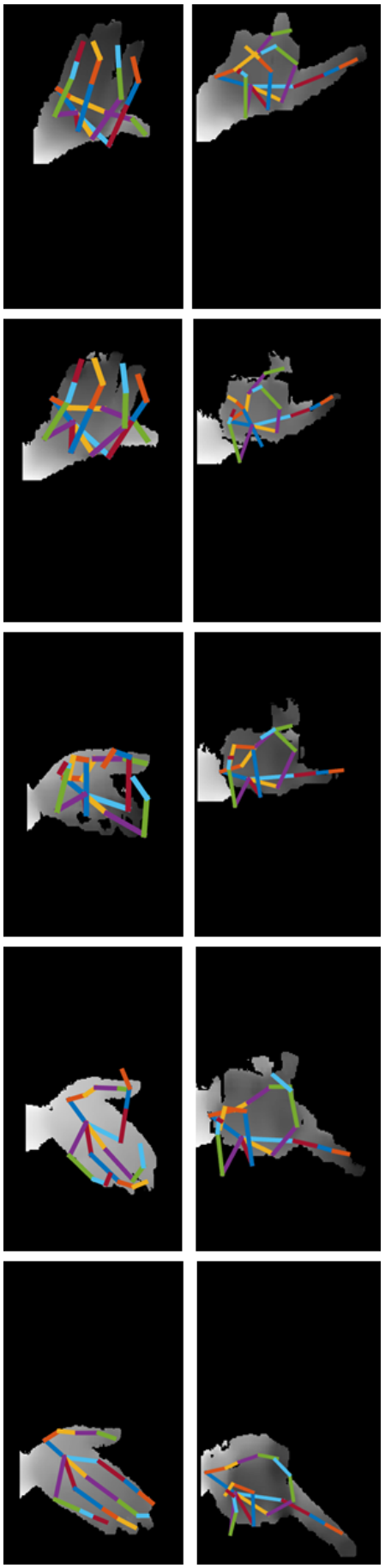


Figure 3.3 – *Depth map sequences. Colored lines represent the hand skeletons extracted from the Intel RealSense [117] camera. Swipe Right gesture performed (top) with one finger and (bottom) with the whole hand from the DHG-14/28 dataset.*

approach. The same thing appears with the pair of gesture *Pinch/Expand* and *Open/Close*. In addition, we supplement this base with the gesture *Grab* because of its usefulness in augmented reality applications, but also to study the scientific challenge of high potentially variations among performers. We also add the gesture *Shake*, as it can be interesting for recognition algorithm to be able to differentiate gesture composed of other gestures (a shake gesture can be seen as a repetition of opposed swipe gestures).

Table 3.1 – *Gesture list included in the DHG-14/28 dataset.*

Gesture	Labelization	Tag name
Grab	Fine	G
Expand	Fine	E
Pinch	Fine	P
Rotation CW	Fine	R-CW
Rotation CCW	Fine	R-CCW
Tap	Coarse	T
Swipe Right	Coarse	S-R
Swipe Left	Coarse	S-L
Swipe Up	Coarse	S-U
Swipe Down	Coarse	S-D
Swipe X	Coarse	S-X
Swipe V	Coarse	S-V
Swipe +	Coarse	S-+
Shake	Coarse	Sh

3.2.3 DHG-14/28 challenges

We focused the creation of the DHG dataset on three main challenges:

- Studying dynamic hand gesture recognition using 3D hand skeletal features;
- Evaluating the effectiveness of the recognition process following the heterogeneity of hand shapes depending on the set of fingers used.
- Distinguishing between both fine-grained and coarse-grained gestures. Indeed, dividing the gesture sequences into two categories – coarse and fine gestures – contributes to increasing difficulties in the recognition challenge.

3.3 Hand gesture recognition using skeletal data

Using 3D hand skeletal data, as depicted in Figure 3.1, a dynamic gesture can be seen as a time series of hand skeletons. It describes motions and hand shapes over the gesture sequence. For each frame t of a sequence, the position in the camera space of each joint is represented by three coordinates, e.g. $j_i(t) = [\ x_i(t)\ y_i(t)\ z_i(t)\ ]$. The skeleton at frame t is then represented by the $3N_j$ dimension row vector:

$$s(t) = [\ x_1(t)\ y_1(t)\ z_1(t)\ ...\ x_{N_j}(t)\ y_{N_j}(t)\ z_{N_j}(t)\ ] \tag{3.1}$$

where N_j is the number of joints which compose the hand skeleton and N_f is the number of frames in the sequence. The final representation of a sequence is a matrix of size $N_f \times 3N_j$ where each line t is the row vector $s(t)$:

$$\mathcal{M} = \begin{bmatrix} s(1) \\ \vdots \\ s(N_f) \end{bmatrix} \tag{3.2}$$

This new data type handles a lot of information on the motion and the shape of the hand over the sequence. In order to fully represent the gesture, we propose to mainly capture the hand shape variation based on skeleton joints, but also the direction of the movement and the rotation of the hand in the space with three distinct features.

3.4 Features extraction from skeletal sequences

In this section, we introduce three distinct frame-wise features computed on hand skeletal data. Two of them aim to represent hand motions, while the last extract hand shape variations though the gesture.

Motion features

Some hand gestures are defined almost only by the way the hand moves in space (e.g. *swipes*). To take this characteristic into account, we compute a direction vector in $\mathbb{R}^3$ for each frame t of our sequence using the position

of the palm joint noted j_{palm}:

$$\overrightarrow{d}_{dir}(t) = \frac{j_{palm}(t) - j_{palm}(t-c)}{\left\| j_{palm}(t) - j_{palm}(t-c) \right\|} \tag{3.3}$$

where c reduce noises when the hand is not moving and is a constant value chosen experimentally. We normalize the direction vector by dividing it by its norm.

For a sequence of N_f frames, we have the set $\mathcal{S}_D$:

$$\mathcal{S}_D = \left\{ \overrightarrow{d}_{dir}(t) \right\}_{[\,1 < t < N_f\,]} \tag{3.4}$$

The rotation of the wrist during the gesture describes also how the hand is moving in space. For each frame t, we compute the vector from the wrist node to the palm node to get the rotational information in $\mathbb{R}^3$ of the hand:

$$\overrightarrow{d}_{rot}(t) = \frac{j_{palm}(t) - j_{wrist}(t)}{\left\| j_{palm}(t) - j_{wrist}(t) \right\|} \tag{3.5}$$

For a sequence of N_f frames, we have the set $\mathcal{S}_R$:

$$\mathcal{S}_R = \left\{ \overrightarrow{d}_{rot}(t) \right\}_{[\,1 < t < N_f\,]} \tag{3.6}$$

Shape features

To represent the variation of the hand shape during the sequence using a full 3D hand skeleton, we propose a descriptor based on sets of joints, denoted as *Shape of Connected Joints* (SoCJ).

Hand skeleton returned from sensors consists of 3D coordinates of joints, represented in the camera coordinate system. Therefore, they vary with the rotation and translation of the hand with respect to the camera. To make our hand shape descriptor invariant to hand geometric transformations, we propose a normalization phase. Firstly, in order to take into account the differences of hand size between performers, we estimate the average size of each bone of the hand skeleton using all hands in the dataset. Secondly, carefully keeping the angles between bones, we change their size by their respective average size found previously. Then, in order to be consistent with the translation and rotation transformations, we

create a reference hand skeleton H_f corresponding to an open hand in front of the camera with its palm node at $[0\ 0\ 0]$. Then, for each frame, we use a *Singular Value Decomposition* to find the optimal translation and rotation from the current hand skeleton to H_f and then apply it to all its joints. This process results in a new hand which keeps its skeleton shape but centered around $[0\ 0\ 0]$ and the palm facing the camera.

Let x represent the coordinates of a joint in $\mathbb{R}^3$ and $T = [x_1\ x_2\ x_3\ x_4\ x_5]$ a tuple of five ordered different joints from the hand skeleton s. To describe the shape of the joint connections, we compute the displacement from one point to its right-hand neighbor:

$$SoCJ(T) = [\ x_2 - x_1\ ...\ x_5 - x_4\ ] \tag{3.7}$$

This results in a descriptor in $\mathbb{R}^{12}$. Figure 3.4 shows an example of a particular SoCJ using the palm's joint and the thumb's.

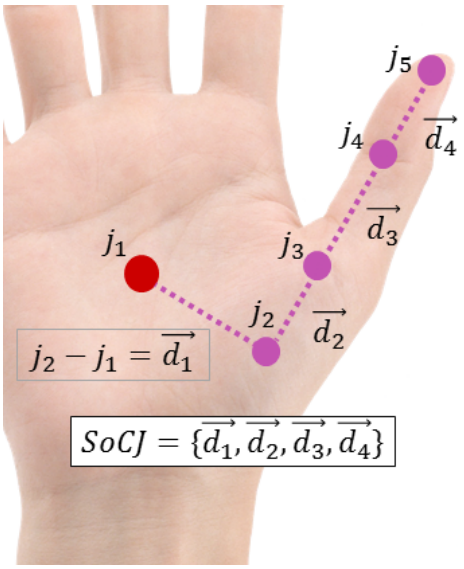


Figure 3.4 – *An example of the SoCJ descriptor constructed around the thumb tuple. Let be* $T = (j_1, j_2, j_3, j_4, j_5)$ *where* $j_i \in \mathbb{R}^3$. *We compute the displacements from points to their respective right neighbor resulting in the SoCJ vector* $\left[\vec{d_1}, \vec{d_2}, \vec{d_3}, \vec{d_4}\right]$.

We remind that the skeleton of the Intel RealSense camera is composed of 22 joints. Theoretically, with $\mathcal{C}$ binomial coefficient function , we can compute $\mathcal{C}(22, 5) = 26334$ different SoCJs for the hand skeleton s resulting

in the set:

$$s_{socj} = \{\ SoCJ(i)\ \}_{[\,1 < i < 26334\,]} \tag{3.8}$$

For a sequence of N_f frames, we have the set $\mathcal{S}_{socj}$:

$$\mathcal{S}_{socj} = \{\ s_{socj}(t)\ \}_{[\,1 < t < N_f\,]} \tag{3.9}$$

3.5 Features representation

The *Fisher Vector* (FV) coding method was first introduced for large-scale image classification. Its superiority against the *Bag-Of-Word* (BOW) method has been analyzed in the image classification [125] as it is going beyond count analysis. It encodes additional information about the distribution of the descriptors. It also has been used over the past five years in action recognition [34, 109, 173, 155].

As a particular hand gesture is so far represented by three sets of features, we aim to use the FV coding method to obtain a statistical representation vector for each of them.

First, we train a K-component *Gaussian Mixture Model* (GMM) using all sets of a particular feature in the training set. We denote the parameters of a GMM by $\lambda = \{\pi_k, \mu_k, \sigma_k\}_{[1 \leq k \leq K]}$ where π_k, μ_k, σ_k are respectively the prior weight, mean and covariance of the Gaussian k. After the training process, we are able to model any new incoming set $\mathcal{S}$ from the test data as follows:

$$p(\mathcal{S}|\lambda) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k p(\mathcal{S}_n|\lambda_k) \tag{3.10}$$

where $\mathcal{S}_n$ is the n^{th} element of $\mathcal{S}$ and N the size of the set. Once we have the set of Gaussian Models, we can compute our FV which is given by the gradient of the formula of Eq. (3.10):

$$\mathcal{G}_\lambda^{\mathcal{S}} = \frac{1}{N} \nabla_\lambda \log p(\mathcal{S}|\lambda) \tag{3.11}$$

The derivatives in Eq. (3.11) are computed separately with respect to mean

and standard deviation parameters, leading to the final FV representation:

$$\Phi(\mathcal{S}) = \{\mathcal{G}^S_{\mu_k}, \mathcal{G}^S_{\sigma_k}\}_{[\,1 \leq k \leq K\,]} \tag{3.12}$$

We also normalize the final vector with a $l2$ and power normalization to eliminate the sparseness of the FV and increase its discriminability. We refer the reader to Sanchez *et al.* [125] for more details.

It is also interesting to notice that the final size of a FV is $2dK$ where d is the size of the feature and K the number of clusters used in the GMM. This observation is a drawback compared to BOW, which has a size of K, when applied to a long descriptor. However, this effect can be ignored in our case where K is relatively small.

3.6 Temporal modeling

The descriptors explained previously in Section 3.4 describe hand shapes and motion variations inside the sequence but they do not take into consideration the dynamic aspect of a gesture. To add the temporal cue, we use a *Temporal Pyramid* (TP) representation already employed in action and hand gesture recognition approaches [34, 175].

The principle of the TP is to divide the sequence into n sub-sequences at each n^{th} level of the pyramid, as depicted in Figure 3.5.

After feature extraction, we represent a sequence of hand skeletons by three sets of different features describing the direction of the hand ($\mathcal{S}_D$), its rotation ($\mathcal{S}_R$) and its shape ($\mathcal{S}_{socj}$) during the sequence. For a new gesture sequence, we use the Eq. (3.12) with the corresponding GMM created from the training set to get the three statistical representation, $\Phi(\mathcal{S}_D)$, $\Phi(\mathcal{S}_R)$ and $\Phi(\mathcal{S}_{socj})$. We compute our three descriptors and their statistical representations for each sub-sequence and concatenate them, as depicted in Figure 3.6. Adding more levels to the pyramid gives more temporal precision but increases the size of the final descriptor and the computing time substantially.

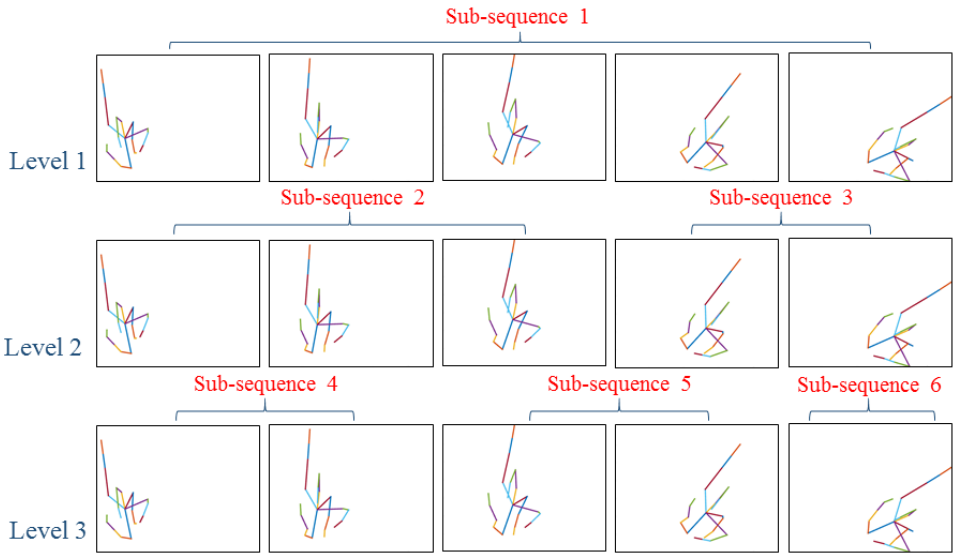


Figure 3.5 – *An example of a Temporal Pyramid of size 3.*

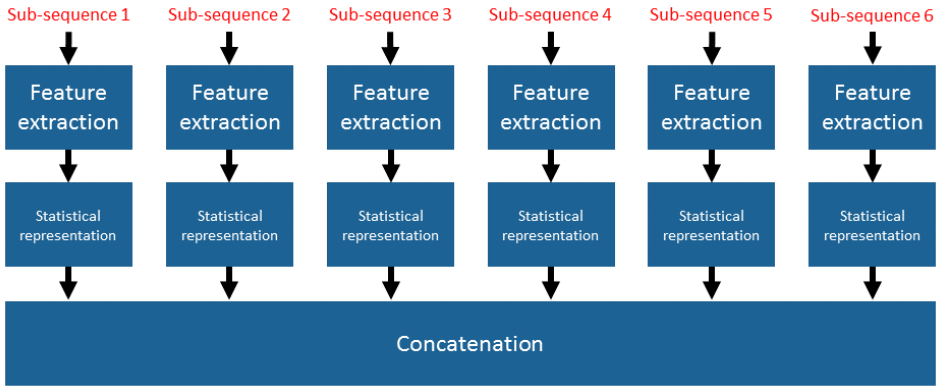


Figure 3.6 – *After temporal modeling, the three descriptors and their statistical representations are computed on each sub-sequences and, finally, concatenated.*

3.7 Classification process

In the field of machine learning, a supervised classification task is the problem of identifying to which of a set of categories – called also classes or labels – a new observation belongs to. The term supervised comes from the idea that the algorithm learns from a training set of data containing observations whose category membership is known.

Let $\mathcal{L} = \{(\beta^i, y^i)\}_{i=1...N}$ be a set of data called a *dataset*, where $\beta^i \in \mathbb{R}^n$ and y^i a categorical variable taking values in $1, \ldots, k$, with k the number of classes. A classification algorithm aim to find a mathematical approximation function $h(\beta^i) : \mathbb{R}^n \rightarrow 1, \ldots, k$ that maps new unknown input data learned from a training set of labeled samples.

In our case, β^i is the concatenation of the three statistical representations of features described in Section 3.4 for each sub-sequences obtained from a pyramid temporal segmentation. Each sequence is also labeled with a gesture type y^i taking values in $1, \ldots, N_g$ where N_g is the number of gesture types in the dataset currently used .

To learn a mapping classification function, we use a supervised learning classifier called Support Vector Machine (SVM) [49]. A SVM model represent the training samples as points in a new space through linear transformations and mapped so that samples of different categories are divided by a clear gap, wide as possible. New incoming unknown samples are then mapped into the same space and predicted to belong to a category based on which side of the gap they fall. If a kernel trick makes possible to add non-linearity inside of the SVM mapping, we do not use it as a linear kernel easily deals with our high-dimensional representation. As originally SVM is a binary classfier, we employ a *one-vs-rest* strategy resulting in N_g binary classifiers. We used the implementation contained in the LIBSVM library [16].

3.8 Experimental results

In this section, we first introduce experimental settings of our method and study intuitive versus automatic selection of features. Second, we evaluate

our proposed approach with two datasets and compare it with four state-of-the-art methods using depth images and skeletal data. We explore its capability to reduce the latency of the recognition process by evaluating the trade-off between accuracy and latency. We also study the impact of the upstream hand pose estimation algorithm on our method, and finally discuss its promising potential and limitations.

3.8.1 Experimental settings

A) Descriptor encoding

We choose the number of levels L_{pyr} of the TP as equal to 4 as it provides a satisfactory compromise between the temporal representation of gestures and the final size of our descriptor. The final size of our computed descriptor is then $(\sum_{i=1}^{L_{pyr}} i) \times (size_{\Phi D} + size_{\Phi R} + size_{\Phi SoCJ})$, where $size_{\Phi x}$ is the FV representation computed from the set of features x. Note that $size_{\Phi} = 2dK$, where K is the number of models created in the GMM. d is the feature dimension: respectively in $\mathbb{R}^3$, $\mathbb{R}^3$ and $\mathbb{R}^{12}$ for the direction, the rotation and the SoCJ features. For FV encoding, we map our descriptors into a K-component GMM with K equal to 8, 8 and 256 gaussians respectively for the direction, the rotation and the SoCJ features. If not mentioned otherwise, we use a *Leave-One-Subject-Out cross-validation* protocol for all conducted experiments.

B) Intuitive versus automatic selection of SoCJ descriptors

On hand skeletal data composed of 22 joints, we can compute 26334 different SoCJs. Using all of them is unnecessary as they provide redundant information and cost computing time. We propose to evaluate two ways to choose our feature set as a combination of the most relevant SoCJs. We firstly evaluate a SOCJ set chosen intuitively and, secondly, by using an automatic suboptimal deterministic feature selection algorithm called Sequential Forward Floating Search (SFFS). In this section, the score $\mathbb{J}$ for each feature set is the classification accuracy obtained using a SVM and only fine gestures of the DHG dataset. We choose this subset of gestures

as the SoCJs are meant to describe the hand shape (50% of the dataset is used for test while using the remainder as training observations).

First, to represent the hand shape, we intuitively divide the hand skeleton into nine tuples of five joints representing the hand's physical structure as presented in Figure 3.7 and from which we compute our SoCJ descriptor. This subset of nine tuples is chosen as it forms a grid on the hand skeleton and each joint appears at least once. This subset obtains a score $\mathbb{J}$ of 73.22%.

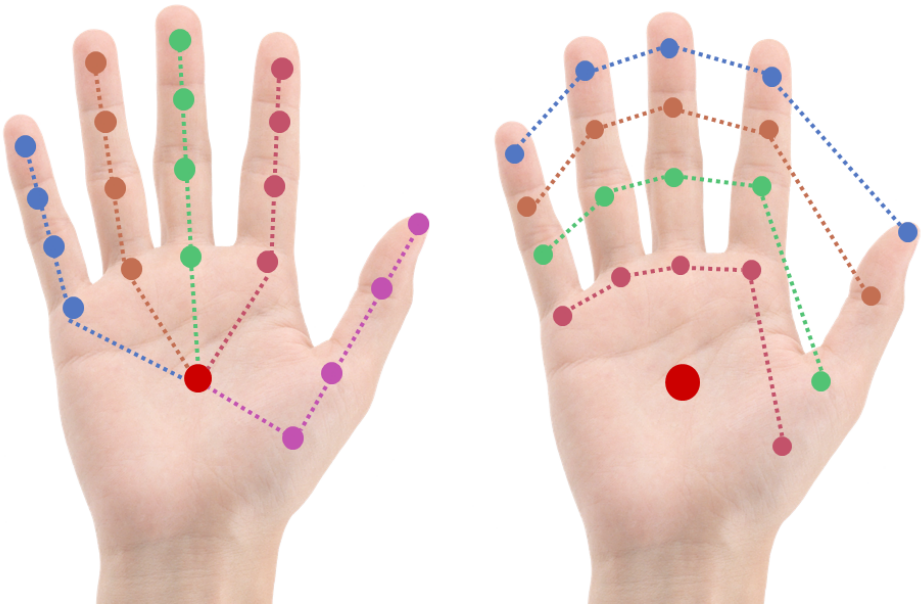


Figure 3.7 – *Nine tuples chosen intuitively to construct the SoCJ descriptors. On the left side are the five constructed with the four joints of each finger plus the palm. On the right side, the one using the five tips, the five first articulations, the five second articulations and the five finger bases.*

Second, we use the SFFS algorithm proposed by Pudil *et al.* [113] in order to automatically choose a relevant subset $\mathbb{X}$ following the score $\mathbb{J}(\mathbb{X})$ laid down above. Starting with an empty set of feature $\mathbb{X}$, this method works in three steps detailed in Algorithm 1.

Results of the SFFS algorithm are shown in Figure 3.8. We obtain a score $\mathbb{J}$ of 75.73% using 10 SoCJs, while adding more SoCJ seems irrelevant as the accuracy does not increase.

Figure 3.9 shows the first three SoCJs selected by the SFFS algorithm.

It is interesting to see that the first one is composed of joints which belong to the thumb and the index, thus providing the necessary information

Algorithm 1: SFFS algorithm

1 $\mathbb{X} = \{\varnothing\}$;
2 $k = 0$;
3 /* Select the best feature. */ ;
4 $x^+ = \arg\max_{x \notin \mathbb{X}_k}(\mathbb{J}(\mathbb{X}_k + x))$;
5 $\mathbb{X}_k = \mathbb{X}_k + x^+$;
6 $k = k + 1$;
7 /* Select the worst feature. */ ;
8 $x^- = \arg\max_{x \in \mathbb{X}_k}(\mathbb{J}(\mathbb{X}_k - x))$;
9 /* Remove the worst feature. */ ;
10 **if** $\mathbb{J}(\mathbb{X}_k - x^-) > \mathbb{J}(\mathbb{X}_k)$ **then**
11 $\quad \mathbb{X}_{k+1} = \mathbb{X}_k - x^-$;
12 $\quad k = k + 1$;
13 $\quad$ Go to 7 ;
14 **else**
15 $\quad$ Go to 3 ;

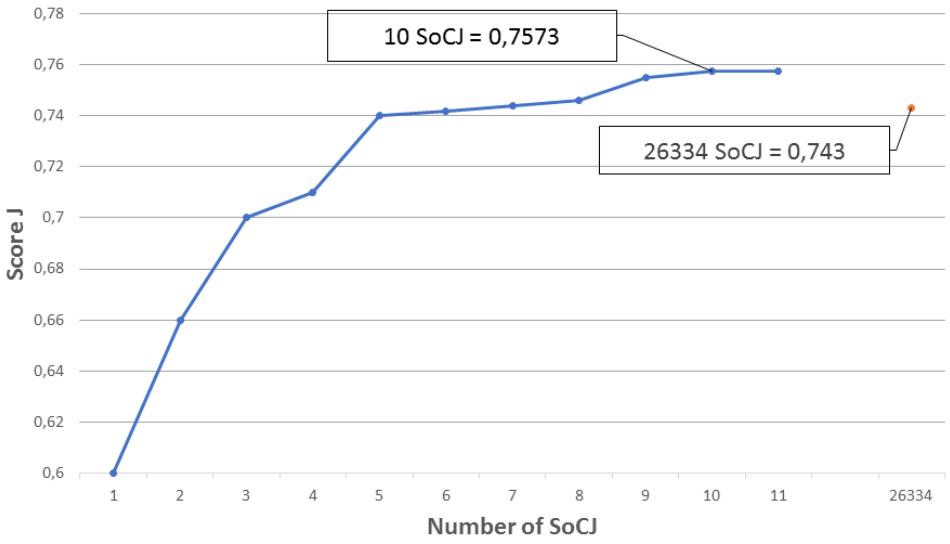


Figure 3.8 – *SoCJ selection using SFFS algorithm on the fine gesture subset of the DHG dataset. The y-axis accuracy is obtained using the number of SOCJ on the x-axis. Using more that 10 SOCJ is not relevant as the accuracy stagnate.*

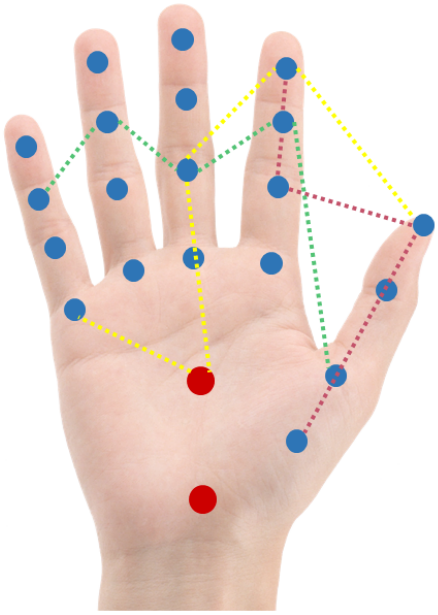


Figure 3.9 – *The first three SoCJ chosen by the Sequential Forward Floating Search algorithm.*

about the hand "clamp". The second one gathers one joint of each finger handling the information of the general shape of the hand (i.e. "open" or "close"). If we use all the 26334 possible combinations, we decrease the accuracy to 74.30%. We understand the drop as a misclassification due to the lack of precision and the redundancy. We note that the computation of 26334 SoCJs for a sequence of 35 frames takes 6.24 seconds, and only 0.0022 seconds for the 10 SoCJs chosen by the SFFS. We use these SoCJs in the following experiments as this subset improves the score compared to the one chosen intuitively.

3.8.2 Hand Gesture Recognition Results

A) DHG 14-28 dataset

To assess the effectiveness of our algorithm to classify gestures of the DHG dataset into 14 classes, we compare the results obtained by the hand shape and motion descriptors separately. Table 3.2 presents the accuracies of our approach obtained using each of our descriptors independently and by combining them. For clarity, we divide the results by coarse and fine

gestures according to the labels from Table 3.1, allowing us to analyze the impact of each descriptor on each gesture category.

Table 3.2 – *Accuracy comparison fine / coarse / both gesture for the DHG-14 dataset.*

Features	Fine (%)	Coarse (%)	Both (%)
Direction	44.60	88.50	72.79
Rotation	50.30	50.61	50.50
SoCJ	67.84	63.12	64.88
SoCJ + Direction + Rotation	74.43	93.77	**86.86**

Using all descriptors (direction + rotation + SoCJ) presented in Section 3.3, the final accuracy of our algorithm on the DHG-14 is 86.86%. It rises to 93.77 % recognition for the coarse gestures, but for the fine ones the accuracy drops below 75%. A large difference can be observed between accuracies obtained for the fine and the coarse gestures, respectively 44.60% and 88.50% when using only the direction. The analysis of the results obtained using only the SoCJ descriptor shows that the hand shape is the most effective feature for the fine gestures with an accuracy of 67.84%. On the other hand, this result shows that the hand shape is also a way to describe coarse gestures with a fair accuracy of 63.12%. If the rotation descriptor shows a low average accuracy of 50.50% for both fine and coarse gestures, it is a valuable feature for pairs of similar gestures such as *Rotation CW* and *Rotation CCW*. These results confirm the interest of using several descriptors in order to completely describe hand gestures.

To better understand the behavior of our approach according to the recognition per class, the confusion matrix is illustrated in Fig. 3.10.

The first observation is that 11 gestures out of 14 have scored higher than 85.00%. The second observation is the great amount confusion between the gestures *Grab* and *Pinch*. By analyzing the sequences, we observe that they are very similar and hard to distinguish even by the human eye. The main difference between them is the hand movement amplitude which is not taken into account by our approach. With a final accuracy of 86.86% obtained on DHG-14 dataset, we noticed that the recognition of dynamic hand gestures is still challenging. The recognition system has

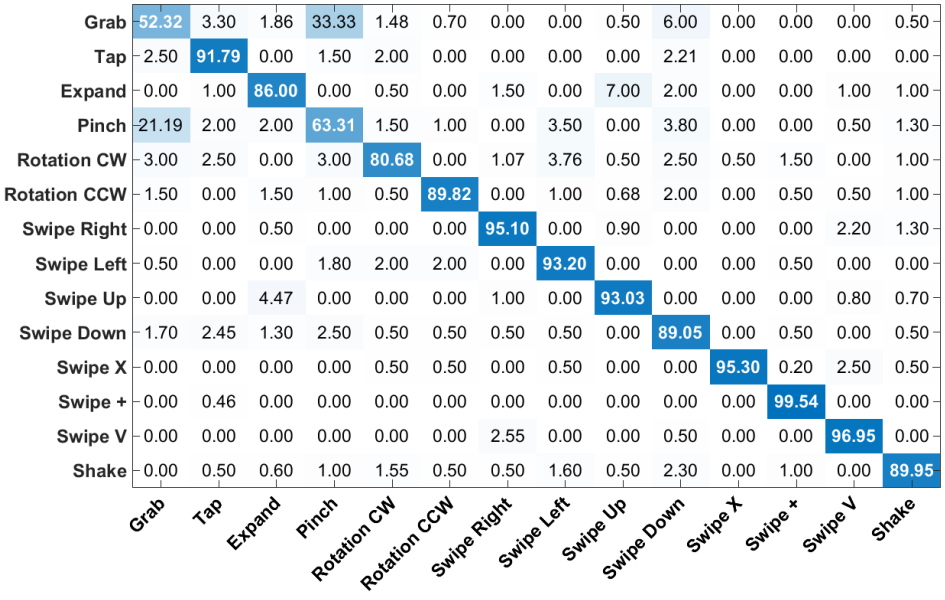


Figure 3.10 – *The confusion matrix of the proposed approach for the DHG-14 dataset.*

to deal with the considerable differences between gestures performed by different people, resulting in a challenging heterogeneity of the gestures.

Finally, in order to meet the challenge of gesture recognition when performed with different numbers of fingers existing in the DHG-28 dataset, we consider hand gestures as belonging to 28 classes related to the gesture type and the way it has been performed (with one finger or the whole hand). The resulting confusion matrix is shown in Figure 3.11. Using our approach, we obtain an accuracy of 84.22%. As shown in Table 3.6, by multiplying the number of classes by two, we lose 2.64% of accuracy.

The impact of using a Temporal Pyramid, a Fisher Vector representation and their combinations in our processing pipeline are also measured and the evaluation results of the DHG-14 dataset are shown in Table 3.3. Note that using SVM directly on features requires the descriptor to have a fixed size. Indeed, we use a linear interpolation to resize the sequence dimension to 35 frames which is the average number of frame of the sequences in the DHG dataset. Obtained results show that adding the FV representation increases the accuracy by 2.87%. The temporal information is very important because it encodes the dynamic aspect of gestures. Taking this into account in our pipeline – by adding the Temporal Pyramid step – increases the final accuracy by 7.10%.

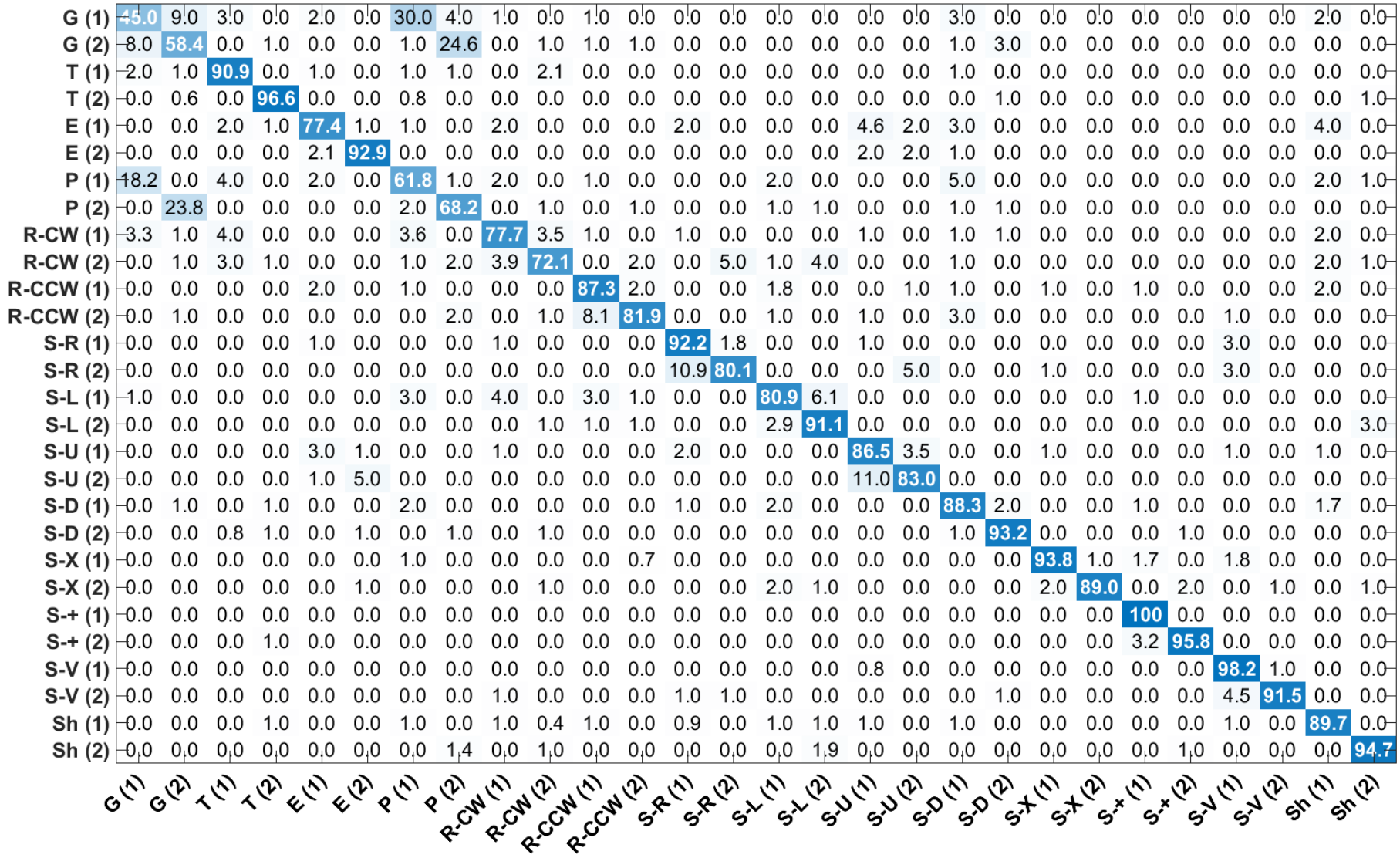


Figure 3.11 – *The confusion matrix obtained by the proposed approach for the DHG-28 dataset. The gestures annotated (1) and (2) were performed respectively using one finger and with the whole hand.*

Table 3.3 – *Recognition accuracies obtained on the DHG-14 gesture dataset following different pipeline configurations.*

Pipeline	Accuracy (%)
Features + SVM	76.89
Features + FV + SVM	79.76
Features + TP + FV + SVM	86.86

B) Handicraft-Gesture dataset

Handicraft-Gesture is a dataset built with a Leap Motion Controller (LMC) [84]. A LMC is a device providing accurate information about the hand skeleton which contains the same 22 joints described in Figure 3.1. This dataset is made of 10 gestures which originate from pottery skills: *poke*, *pinch*, *pull*, *scrape*, *slap*, *press*, *cut*, *circle*, *key tap*, *mow*. The data are captured at a rate of 60 frames per second. There were 10 volunteers helping to build the dataset and each one performed every gesture three times. Therefore, the Handicraft-Gesture dataset contains 300 sequences of dynamic hand gestures.

To evaluate our approach on the Handicraft-Gesture dataset, we follow the experimental protocol proposed by Lu *et al.* [84], i.e. *Leave-One-Subject-Out* cross-validation. They compute several features based on palm direction, palm normal, fingertip positions, and palm center position. For the classification of temporal sequences, they use a Hidden Conditional Neural Field classifier. Table 3.4 shows how the hand gesture recognition accuracy has been increased by 2.11% using our approach.

Table 3.4 – *Recognition accuracies obtained on the Handicraft-Gesture dataset.*

Method	Accuracy (%)
Lu *et al.* [84]	95.00
Ours	**97.11**

In order to investigate the impact of each of our descriptors (direction, rotation and SoCJ) on the Handicraft-Gesture dataset, Table 3.5 presents the accuracy obtained using each of those descriptors independently and by concatenating them in only one descriptor. The direction and the rotation of the hand through the movement give acceptable results, respectively 71.66% and 62.67%. However, the score increases to 92.35% using

only the SoCJ descriptor. Indeed, pottery skills require fine hand gestures which do not contain a lot of motion information, similarly to fine gestures of the DHG dataset. These gestures are better described by hand shape variations, so, that is the reason why the SoCJ is considered as the most effective feature for recognition.

Table 3.5 – *Recognition accuracies obtained on the Handicraft-Gesture dataset for each descriptor of our approach.*

Features	Accuracy (%)
Direction	71.66
Rotation	62.67
SoCJ	92.35
SoCJ + Direction + Rotation	97.11

We also note, as shown in Table 3.2, that combining the descriptors leads to a significant gain in performance. This combination is more useful for the DHG dataset that for the Handicraft-Gesture dataset where adding the motion features improves the recognition rate by only 4.76%. That is explained by the different nature of gestures incorporated in the datasets. Gestures of the Handicraft-Gesture dataset come from pottery skills. The DHG dataset is more heterogeneous as it also contains coarse gestures for which our motion features are important. In fact, both coarse and fine gestures are useful in a Human-Computer Interface. Future gesture recognition algorithms will have to take the two varieties into account.

C) Comparison with state-of-the-art methods

We compare our approach with four state-of-the-art methods on the DHG dataset. We chose two depth-based descriptors: HOG^2 proposed by Ohn-Bar *et al.* [104] and HON4D proposed by Oreifej *et al.* [107]. We also compare our approach to a skeleton-based method proposed by Devanne *et al.* [27] showing a good accuracy for human action recognition. Devanne's approach is based on a similarity metric of human trajectories using the shape of 3D body skeleton in a Riemannian manifold. Finally, we compare the hand shape descriptor SoCJ with a similar state-of-the-art

feature called Skeletal Quad defined by Evangelidis *et al.* [34]. The publicly available source codes of these methods are used in our experiments.

For the two depth-based descriptors [104, 107], pre-processing steps on the depth sequences are needed. First, using a suitable threshold, we clean the image by removing the background and the body of the subject keeping only the region-of-interest of the hand. Then, we crop the size of the images by removing all regions where the hand does not appear along the sequence. For the HON4D method, we choose a spatio-temporal grid of size $5 \times 5 \times 3$ since it gives the best accuracy. For Evangelidis *et al.* method [34], in order to properly compare the hand shape descriptors, we use our approach by swapping our SoCJ with the Skeletal Quad descriptor while keeping the rotation and direction features.

Table 3.6 presents the results obtained by the methods cited previously using 14 and 28 gestures on the DHG dataset. We note that our approach outperformed, with an accuracy of 86.86%, the two depth-based descriptors showing the promising direction of using skeletal data for hand gesture recognition. The accuracy obtained by the action recognition method [27] applied for 3D hand joints trajectories is 76.61%. It shows that an action recognition approach is often not appropriate for hand gesture recognition and that hand trajectories are not sufficiently distinctive enough for hand gesture classification.

Table 3.6 – *accuracy comparison 14 / 28 gestures for the DHG dataset.*

Method	14 gestures (%)	28 gestures (%)
Ohn-Bar *et al.* [104]	81.85	76.53
Oreifej *et al.* [107]	75.53	74.03
Devanne *et al.* [27]	76.61	62.00
Evangelidis *et al.* [34]	84.50	79.43
Ours	**86.86**	**84.22**

When we apply these methods on 28 classes, the HOG^2 descriptor [104], which had a good result on 14 gestures, obtains 76.53% of accuracy. The depth-based methods do not handle enough hand shape information to deal with the challenge of hand gestures performed with different numbers of fingers. We note that Devanne's approach loses 14.61% of recognition rate on this experiment showing that the method, giving a good

result on action recognition dataset, it is unsuitable for fine and dynamic hand gesture recognition.

Evangelidis *et al.* [34] propose a local body skeleton descriptor that encodes the relative position of joint quadruples. It requires a *Similarity Normalisation Transform* (SNT) that leads to a compact (6D) view-invariant skeletal feature, called Skeletal Quad. Because of the SNT, their descriptor takes more computation time and is less suitable for hand shape description as it lost information about distances between joints. The accuracy on the DHG-28 dataset using their hand shape descriptor decreases by 4% compared to the SoCJ descriptor.

In Table 3.7, we investigate the impact of the different methods on the fine and coarse gestures separately. We notice that coarse gestures are defined by the motion of the hand in space and fine gestures are more distinguished by the variation of the hand shape during the sequence. The statement of a need of precision in the field of dynamic hand gesture recognition is also shown in this experiment. Except for the HOG^2 descriptor [104], Oreifej *et al.* [107] and Devanne *et al.* [27] give honorable results in the task of coarse gesture classification but they show a lack of precision generating a recognition rate below 61% when trying to classify fine gestures. Although our approach gives the best results with 74.43% of correctly labeled fine gestures, we note that further improvements are needed.

Table 3.7 – *Accuracy comparison for coarse / fine gestures of the DHG-14 dataset.*

Method	Coarse gestures (%)	Fine gestures (%)
Ohn-Bar *et al.* [104]	86.00	71.60
Oreifej *et al.* [107]	83.88	60.50
Devanne *et al.* [27]	86.61	58.60
Evangelidis *et al.* [34]	92.22	70.62
Our approach	**93.77**	**74.43**

3.8.3 Latency analysis and computation time

For many applications, making a potentially unreliable forced decision based on partial available frames is a real challenge. The goal of the following experiments is to automatically determine when a sufficient num-

ber of frames are observed to provide a reliable recognition of the occurring gesture, hence the term *low-latency* recognition. The *latency* can be defined as the time lapse between the moment when a sequence is given to the algorithm and the instant when the system recognizes the performed gesture. We study here two characteristics: computational and observational latency. The computational latency is the time the system takes to perform the recognition process. The observational latency represents the percentage of a continuous gesture needed by a system in order to perform its recognition.

A) Computational latency

The computational time is a very important characteristic of a hand gesture recognition algorithm as it should be working in real time for some HCI applications. We evaluate the computational latency of our approach on the DHG-14 dataset, using a MatLab implementation with an Intel Xeon CPU E3 3.40 GHz and 8 GB RAM. Since the proposed approach is based only on skeletal joint coordinates, it needs a small computation time. Table 3.8 reports the minimum, average and maximum computation time for each step of our approach. For the whole recognition process, the average computation time is 0.2502s for a sequence of 35 frames. This time makes our approach suitable for real-time recognition. We note that 88.49% of this time is taken by the classification process.

Table 3.8 – *Computation time in second for each step of our approach on the DHG-14 dataset. We note that some steps are dependent of the size of the sequence. We report the time for the smallest sequence (N_f = 20), the mean size over all the sequence (N_f = 35) and the biggest sequence (N_f = 150).*

Step	Mins (sec)	Averages (sec)	Maxs (sec)
Normalization of hand size	0.0038	0.0154	0.0640
Direction descriptor	0.0002	0.0011	0.0045
Rotation descriptor	0.0001	0.0005	0.0026
Registration of the hand	0.0009	0.0038	0.0157
SoCJ descriptor	0.0005	0.0022	0.0089
FV and TP construction	0.0033	0.0058	0.0188
Classification	0.1905	0.2214	0.2150
Total	**0.1993**	**0.2502**	**0.3295**

B) Observational latency

To analyze the observational latency of our approach, we show how the accuracy depends on the percentage of the sequence. New recognition rates are computed by processing only a percentage of the sequence length. In each case, we cut the training sequences into shorter ones to create a new training set. During the classification step, we also cut the test sequences to the corresponding length and apply our method with the same learning protocol *Leave-One-Subject-Out* cross validation.

Figure 3.12 shows the observational latency of our approach on the DHG-14 and the Handycraft dataset. We see that a near-maximum accuracy is obtained using 60% of each sequence on both datasets. In other words, the evaluations in terms of latency have revealed the efficiency of our approach for rapid gesture recognition. It is possible to recognize a gesture from the DHG-14 dataset composed of 50 frames up to 80.82% seeing only 30 frames (versus 86.86% using all the frames). Thus, our approach can be used for interactive systems, notably, in entertainment applications to resolve the problem of lag and improve some gesture-based games.

This shows that the computational latency can be masked by the observational latency in the cases where sequences are nearly twice as long as the computational latency.

3.8.4 Influence of the upstream hand pose estimation step on hand gesture recognition

Hand pose estimation is a growing field of research. The needs of precise mid-air HCI for emerging applications (i.e. interaction with a virtual or augmented realty world) has attracted much attention in the Computer Vision community [101, 40, 75, 76, 140]. We conduct a final experiment in order to assess the influence of the depth-based hand pose estimation on our approach.

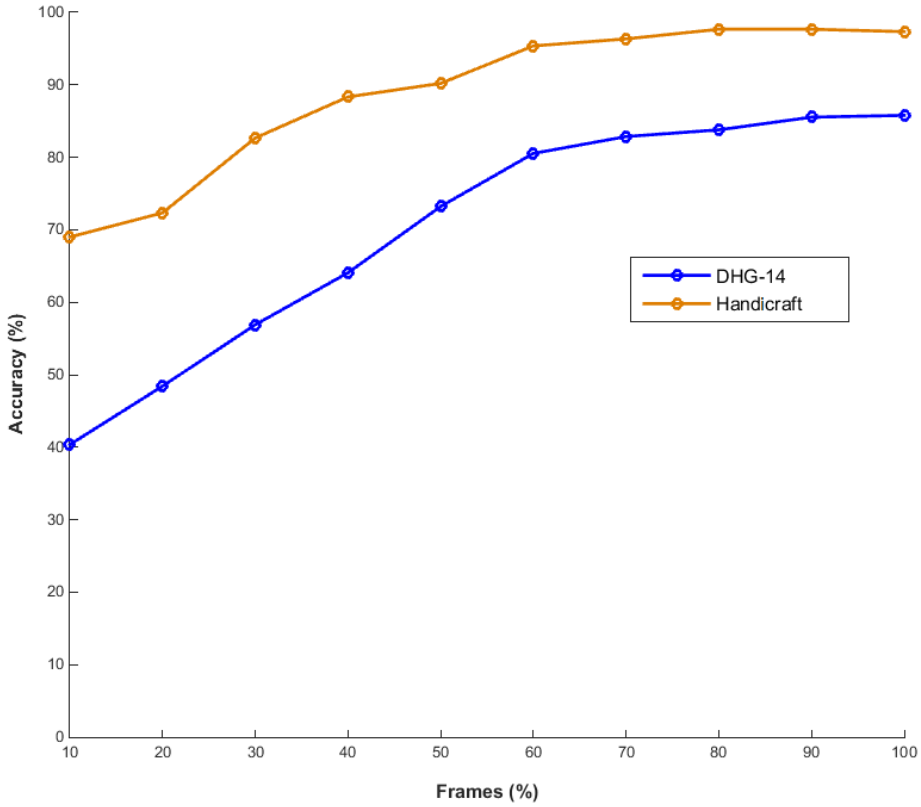


Figure 3.12 – *Observationnal latency analysis on the DHG-14 and Handicraft-Gesture datasets. The accuracy on the y-axis is obtained by processing only the percentage of the sequence shown in the x-axis.*

A) Hand Pose Estimators

In order to measure the impact of hand pose estimation on gesture recognition, we evaluate three distinct hand pose estimators on the DHG-14 dataset: the methods proposed by Oberweger *et al.* [101] and Ge *et al.* [40] in addition to the Intel RealSense estimator [117]. The hand pose estimator proposed by Oberweger *et al.* in [101] predicts joint positions from hand depth images using Convolutional Neural Networks (CNN). For the CNN training step, we use the ICVL dataset [144] composed of 180,000 ground truth annotated depth images with the 3D joint locations of the hand. The depth images comes from the Intel Creative depth sensor. The second method proposed by Ge *et al.* [40] projects the hand depth images onto three orthogonal planes and utilizes these multi-view projections to regress into 2D heat-maps using CNNs which estimate the joint positions on each plane. These multi-view heat-maps are then fused to produce final 3D hand pose estimation. The dataset, introduced by Sun *et al.* [140] and composed of 76500 depth images captured using the Intel Creative depth sensor, is used to train the CNNs. A cross-dataset challenge has been experimented by Ge *et al.* [40] to verify the possible generalization of their method by showing its ability to use a dataset for training and another one for testing.

We use in these experiments the region-of-interest of the hand returned by Intel RealSense depth camera as inputs to pre-trained hand pose estimator algorithms instead of a particular hand extraction algorithm, without any preprocessing step.

Finally, we perform hand gesture recognition using our method from the estimated 3D joint positions obtained by each one of the two estimator algorithms, and compared the results with those obtained using the Intel RealSense. Figure 3.13 shows the recognition accuracies on our DHG-14 dataset per class of gestures. The average accuracies by estimator, available in Table 3.9, show that the performance of our method is independent of the pose estimation algorithm.

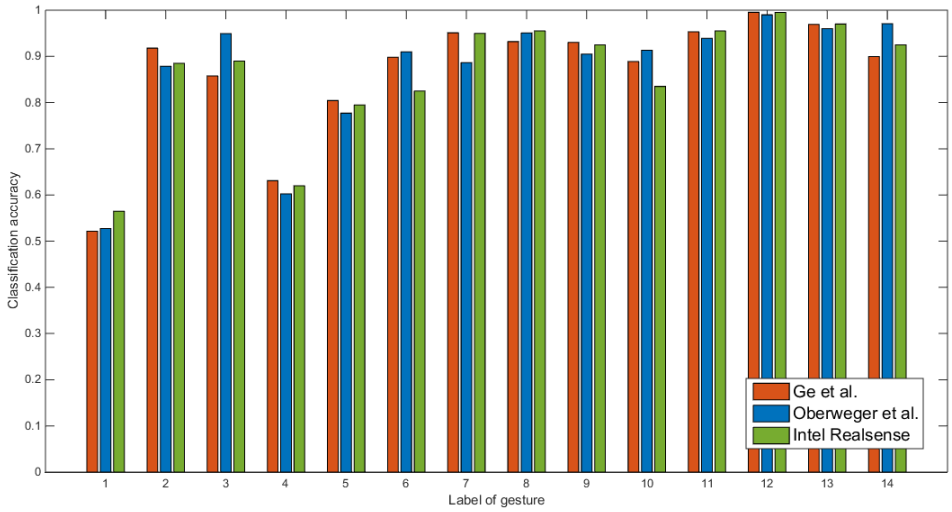


Figure 3.13 – *Recognition accuracies per class of gesture on the DHG-14 dataset using three hand pose estimators.*

Table 3.9 – *Average recognition accuracies obtained on the DHG-14 dataset using three hand pose estimators.*

Hand pose estimation algorithm	Accuracy (%)
Ge *et al.* [40]	86.92
Oberwerger *et al.* [101]	86.24
Intel Realsense [117]	86.86

B) Results on the NVIDIA Dynamic Hand Gesture dataset

Recently, Molchanov *et al.* [94] introduced a new challenging multimodal dynamic hand gesture dataset captured with depth, color and stereo-IR sensors in a car simulator. Using multiple sensors, they acquired a total of 1532 gestures of 25 hand gesture types as depicted in Figure 3.14. A total of 20 subjects participated in data collection, performing gestures with their right hand. The SoftKinetic DS325 sensor is used to acquire frontal view color and depth videos. We evaluate our approach applied to this challenging dataset, using Ge *et al.* hand pose estimator [40] which gives the best recognition accuracy on DHG-14 dataset (see Table 3.9). The extracted 3D joint positions of hand from depth images are used as input for our gesture recognition method. Following the same protocol proposed in [94], we randomly split the data into a training (70%) and a test (30%) sets, resulting in 1050 training and 482 test videos. We perform the hand region-of-interested extraction step using the same algorithm used

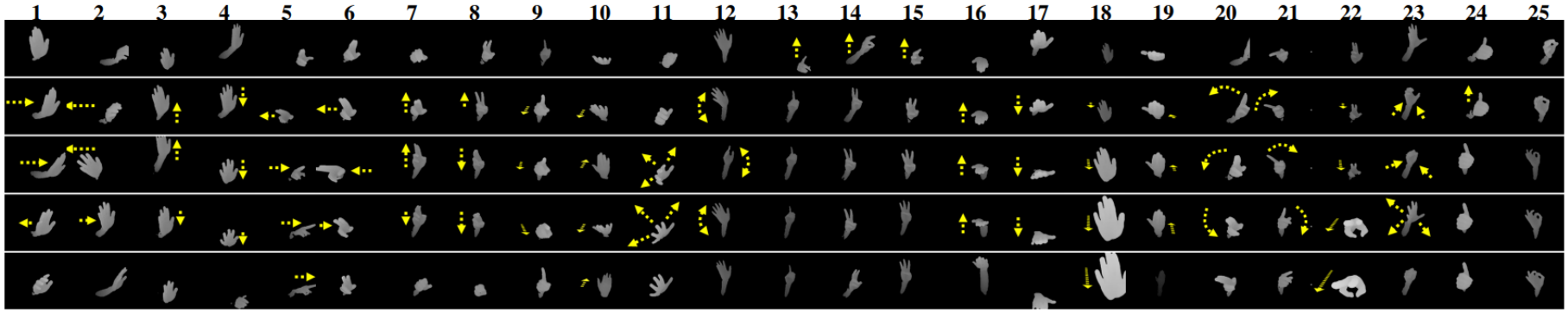


Figure 3.14 – *The 25 different gestures of the NVIDIA hand gesture dataset. Each column shows a different gesture class (1-25). The gestures included (from left to right: moving the hand left, right, up, or down; moving two fingers left, right, up, or down; clicking with the index finger; calling someone (beckoning with the hand); opening and shaking the hand; showing the index finger, two fingers or three fingers; pushing the hand up, down, out or in; rotating two fingers clockwise or counter-clockwise; pushing forward with two fingers; closing the hand twice; and showing "thumb up" or "Ok". The top and bottom rows show the starting and ending depth frames, respectively, and yellow arrows indicate the motion performed in intermediate frames. Image reproduced from [46].*

by Oberwerger *et al.* and Get *et al.* [101, 40]. In Table 3.10, we compare our approach to two handcrafted methods (HOG+HOG^2 [105] and Super Normal Vector (SNV) [167]) and two deep learning methods (C3D [149] and R3DCNN [94]). It should be noticed that we show results obtained by the state-of-the-art methods using only depth information. The recognition accuracies per gesture types obtained by our approach and by the R3DCNN method [94] are also presented in Fig. 3.15.

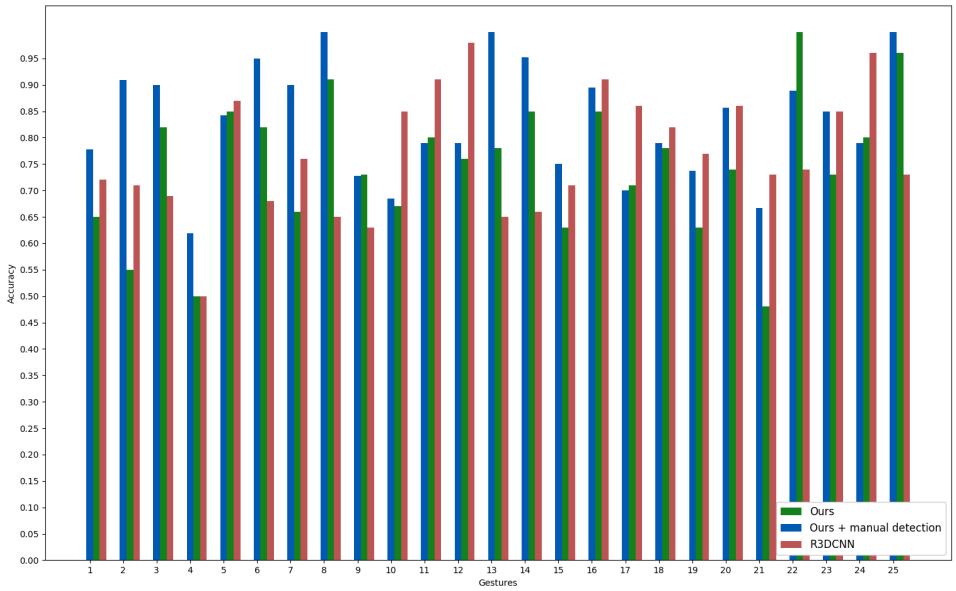


Figure 3.15 – *Comparison of recognition accuracies per class of gestures on the NVIDIA Dynamic Hand Gestures dataset. The list of gestures – annotated 1 to 25 – is depicted in Figure 3.14.*

Table 3.10 – *Comparison of our method to the state-of-the-art methods on depth images of the NVIDIA Dynamic Hand Gesture dataset.*

Method	Type	Data	Accuracy (%)
HOG+HOG^2 [105]	Hand-Crafted	Depth	36.3
SNV [167]	Hand-Crafted	Depth	70.7
C3D [149]	Deep Learning	Depth	78.8
R3DCNN [94]	Deep Learning	Depth	80.3
Ours	Hand-Crafted	3D Hand Skeletal	74.0

Our results allow us to make three main observations: First, with a final recognition accuracy of 74.0%, we go beyond the two handcrafted methods [105, 167] which use depth image based descriptors and obtain respectively 36.3% and 70.7%. Second, deep learning methods outperformed

recent results in many domains in computer vision. Following this statement, 3D convolutional layers presented in [149, 94] show particularly reliable accuracies on the task of 3D hand gesture recognition, obtaining, respectively, 78.8% and 80.3%. The R3DCNN method [94] gets better results using a recurrent layer after the 3D convolution in order to model the global temporal information.

Finally, despite the overall superiority of the R3DCNN method, our approach provides more accurate recognition accuracy for eight gestures: hand up (69% to 80%), two fingers left (68% to 80%), two fingers down (65% to 89%), showing one (65% to 78%) or two (66% to 85%) fingers, one (63% to 73%) or two (74% to 100%) fingers forward , "OK" sign (73% to 94%). Labels of these gestures in Figure 3.14 are, respectively, 3, 6, 8, 13, 14, 9, 22 and 25.

It is interesting to note that the R3DCNN method outperforms our approach mostly on gestures with an open hand and where the gesture can be described mainly by the movement in space. In contrast, our method surpasses the recognition of the different number of fingers used during the sequence. We interpret this as our approach lacking precision to describe the dynamic part of hand gestures. It remains to be noted that the hand pose estimator still has difficulty to estimate the hand pose in certain acquisition conditions (*i.e.* as the the hand is perpendicular to the camera or when the user performs a quick gesture). In fact, hand pose estimation is a challenging task and recent state-of-the-art approaches still need improvements. We expect enhancing hand pose estimation will bring improvement of hand gesture recognition.

Moreover, the R3DCNN is composed of a large neural network with more than 50 million parameters. They use a NVIDIA DIGITS DevBox with four Titan X GPUs to train and predict a new incoming gesture. Currently, this configuration is not suitable for real applications. Additionally, with the use of our SoCJ descriptors, we are able to analyze the impact of our descriptors on the recognition process(3.8.1), which is not possible using a deep neural network refered to as a “black box”.

3.9 Conclusion

In this chapter, we explored a way to perform dynamic hand gestures recognition using hand skeletal features. We proposed a method using three gestural features: hand's direction, rotation and a set of Shape of Connected Joint to extract an efficient hand shape information. Each set of features is transformed in a statistical representation using a Fisher Kernel. In addition, we computed those descriptions on overlapping subsequences of gestures using a temporal pyramid representation. The evaluation of our approach shows a promising way to perform hand gesture recognition using skeletal-based features.

Experiments are carried out on three hand gesture datasets, containing a set of fine and coarse heterogeneous gestures captured in different scenarios. Furthermore, results of our approach in terms of latency demonstrated improvements for a low-latency hand gesture recognition systems, where an early classification is needed. Comparative results with state-of-the-art methods demonstrate that our approach outperforms existing handcrafted approaches.

Moreover, we also revealed a lack of precision to describe the dynamic of complex hand gestures, compared with the feature learning power of modern deep learning models.

In the two following chapters, we focus on deep learning strategies in order to better represent the complex dynamic and temporal information of hand gestures. Moreover, gesture detection in an online scenario are considered as an extension of our current approach.

4

Recent deep learning approaches in Computer Vision

Contents

4.1 Introduction

Recently, many applications of the Computer Vision (CV) field shown a change of paradigm. From human activity recognition to speech recognition, image classification and labeling, CV areas see the emergence and successful arrival of the machine learning technology called deep learning. Since 2010, researchers migrate from traditional handcrafted features to learned-based features also called data-driven algorithm. There are many learned-based feature methods for vision recognition tasks such as dictionary-based approaches or genetic programming. Nevertheless, we focus on deep learning as, in recent years, it changes the game in computer vision.

4.1.1 Different pipelines in Computer Vision: handcrafted versus deep learning approaches

Vision-based recognition algorithms can be divided into two categories: handcrafted and learned-based feature methods. Many CV challenges can be solved by handcrafting the right set of features from the data to accomplish the task. The pipeline of handcrafted based algorithms consists often of three main steps:

1) Data generation and pre-processing. Data are captured from devices, e.g. 2D and/or 3D sensors, which are inputs of algorithms. Raw data are often pre-processed. Generally, the pre-processing step consists of foreground detection, background removal and/or raw data filtering to remove outliers and defaults. Often, inputs can also be a representation *computed* from the original data. For example, a set of 3D joints – called a body skeleton – is retrieved and computed from a sequence of human depth images and is the input of a gesture recognition algorithm [27].

Algorithms take generally a combination of multiple raw data and/or representations as inputs in order to get a maximum of relevant information [99].

2) Handcrafted feature extraction. Handcrafted features is also called feature engineering. A handcrafted feature comes generally from human intuitions and problem-specific prior knowledge. For researchers, it consists of understanding a problematic emerging from a specific type of data, e.g. how to recognize different hand gestures from depth images? As computers deal with numbers, researchers extract a bunch of mathematical features from raw data that correctly help to solve the problematic.

3) Trainable classifier. Machine learning allows us to tackle tasks that are too difficult to solve with a hand design program. *Classification* problems are machine learning tasks where the program is asked to specify which of K categories or labels some input data belongs to.

Let $\mathcal{L} = \{(\beta_i, y_i)\}_{i=1...N}$ be a set of data called a *dataset*, where $\beta_i \in \mathbb{R}^n$ and y_i a categorical variable $\in 1, \ldots, k$ called *label*, such that $y_i = f(\beta_i)$ known as *ground-truth*. The goal here is to find a mathematical approximation function $\hat{y}_i = h(\beta_i, \theta)$ of f that maps the input data with its label. This function – called a *classifier* – has to minimize a *cost function* which penalizes the mismatching between the output of the classification function $\hat{y}$ and the ground-truth y.

Feature engineering is able to incorporate human ingenuity and problem-specific prior knowledge. However, the lack of knowledge and the high level of abstraction of the task make difficult to find a correct mathematical representation of the data. One possible solution of this problem is to use machine learning to discover not only the classification mapping but also the efficient set of features from the original data. In the deep learning paradigm, as show in Figure 4.1, we replace handcrafted features by computing low and abstract features using a learning algorithms.

4.1.2 Feature extraction

Pre-processing, representation and feature engineering are steps of the recognition process that happen after capturing raw data and before using a classifier. They can be grouped in a step called *feature extraction*.

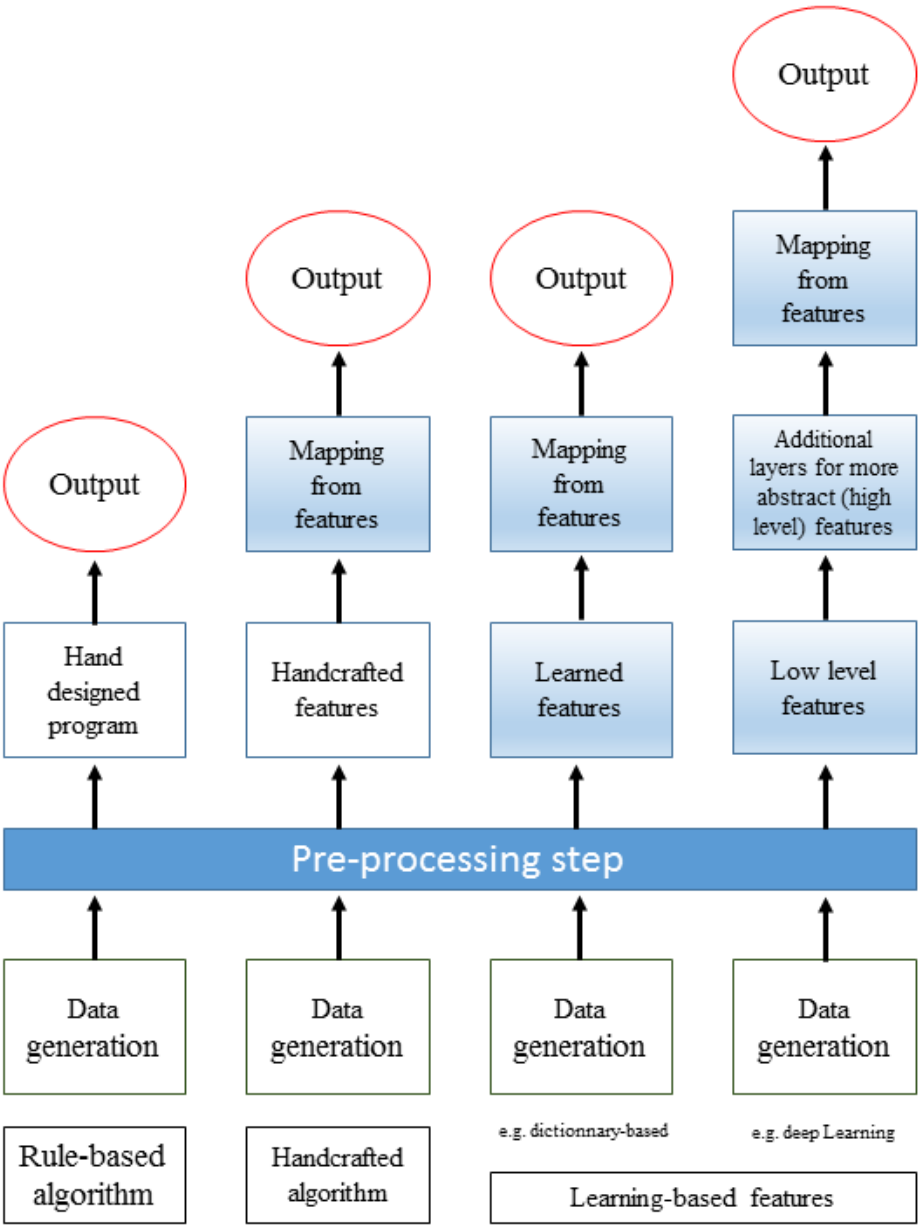


Figure 4.1 – *Flowcharts showing how the different parts of the AI system relate to each other within different strategies. White squares indicate operations that are able to learn from data. Blue squares indicates hand made operation. Green squares and red circles represent, respectively, data and label.*

Boundaries between steps of pre-processing, representation, and features engineering are unclear. Bengio *et al.* [9] use the words *representation* and *feature* as synonyms in their survey in the area of unsupervised feature learning and deep learning. For example, are data normalization and standardization pre-processing steps or a new representation of data in a different mathematical basis? The common idea behind the difference between a representation and a feature is the possibility of performing the inverse function if using a representation.

Figure 4.2 is depicting a simple example of the impact of data representation on the classification process. We display two sets of 2D points in the Cartesian space where one is circled by the other. If we aim to separate the two sets by a single line, the task seems impossible. Unless, we represent the points in the Polar space and, thus, easily separate the two sets by a straight vertical line.

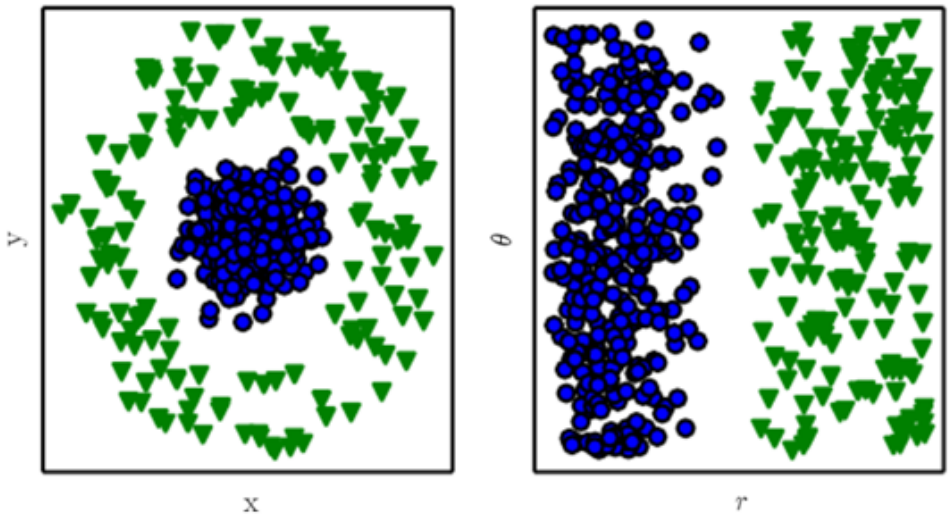


Figure 4.2 – *Example of different data representation. Let us suppose we want to separate by a line the two set of 2D points in the Cartesian space coordinates displayed on the left picture. This task is impossible. By passing into the Polar space coordinates showed in the right picture, the task becomes simple to solve with a vertical line. Image reproduced from [41].*

Theoretically, a fully data-driven algorithm should take as inputs raw data but none works in that way. A pre-processing step is always necessary and a representation is often used. Some of the best performing deep neural networks on recognition tasks still use representation or even engineered features as inputs. Recently, Jaderberg *et al.* [56] proposed to

learn – instead of compute – the pre-processing step, trying to make their algorithms real fully end-to-end data-driven algorithms.

4.1.3 Pros and cons

Performances of traditional machine learning algorithms, such as Support Vector Machine [49], Ramdom Forest [12] or Hidden Markow Model [116], heavily depend on the chosen data representation. However, handcrafted features often suffer from a loss of information.

Deep Learning algorithms have recently given particularly amazing results on many challenges in CV but they also suffer from drawbacks. They need a huge amount of data to work properly, which is still a problem in some area of research where data are not created easily such as 3D data. Additionally, training and parameterizing deep neural networks need a lot of computation resources and experimentation times.

Are handcrafted algorithms will become obsolete? It yet exists a wide area of applications that needs handcrafted features. To better understand when and where we should use one or the other, let us have a look at two practical examples:

First example. A group of researchers have a dataset of 200,000 images representing random colored geometric forms. A psychological experiment have been done and each of these images have been binary classified according to whether children like them or not. We know the images are labeled correctly. However, psychologist did not find explanations about why children like an image and not another. Whatever the reasons, researchers aim to classify an incoming image according to whether it will please a child or not.

This is a perfect challenge to use deep learning algorithm. First, researchers have a huge amount of labeled data. Second, experts have no idea about how to extract effective features from data in order to solve the problem. In this case, we can use a deep neural network in order to tackle the challenge.

Second example. A group of researchers have a dataset of 1000 images of a rare skin disease. Physicians classified them as *sick* or *not sick*. They aim to develop an algorithm that is able to classify a new incoming skin image. The application aims to be used in old computers with low computational resources.

In this case, we miss data in order to use a deep learning algorithm. Additionally, deep neural networks, as Convolutional Neural Network (see 4.4.3), need a powerful hardware configuration to works on images. Compared to learned features, this case will profit the flexibility and computational efficiency of features engineering, and it does not rely on a (too) large dataset.

Deep neural networks are a revolution and a wonderful technology that already prove their effectiveness. However, fully end-to-end learned deep neural network models, often referred to as "black-box", have limitations. Sometimes, using deep neural networks as the only hammer might make the solution over-engineered. It is also difficult to take into account problematic-specific prior knowledge into deep neural network models. Finally, once features are learned they are hardly understandable by humans.

4.2 Where does deep learning come from and why is it so hot topic right now?

4.2.1 History

Since ancient Greece, humanity dreams about creating new forms of intelligence. In his book, *Metamorphoses*, Ovid, one of the most famous ancient Roman poet, told the story of *Pygmalion*, an artist which, with the help of gods, gave life to a statue he had carved. In the Jewish folklore exists the *Golem*, a half-intelligent creature made of mud and magic.

In 1842, the Countess of Lovelace, Augusta Ada King Noel wrote the first algorithm which made her the first programmer in the history. Since then, computer technology did not stop to evolve, making objects more and more intelligent. Artificial Intelligence (AI) is one of the most active

topic of research in computer science and have many practical applications. Yesterday, we asked computers to perform routine labor for us. Today, we ask them to understand speech, images and videos, or yet, to help doctors to diagnose diseases.

In AI, the big question is: how to make a computer learn by itself ? As we saw in the previous section, the traditional way to do it is to find an expert of the topic you want the computer to learn about. With its problem-specific prior knowledge, you are able to write a rules based program that makes the computer helpful. What makes deep learning really interesting is that it do not need a deep implication from experts about a specific problem to learn a possible solution. One thing that need to keep in mind is that, so far, we still need label data and human intuitions to find an efficient objective function.

As early as 1943, Warren McCulloch and Walter Pitts introduced a model of neurological networks. They recreated neurons based on threshold switches and showed that even simple networks of this kind are able to calculate any logic or arithmetic function [90]. In the 1950s, following their statements and inspired by the successfully working brain systems and its wonderful capability to learn, Frank Rosenblatt developed the idea of an artificial neuron called *Perceptron* [122].

4.2.2 Perceptrons and biological neurons similarities

Biological neurons. Many things are still unknown about how the brain trains itself. In the human brain, a neuron collects electrical signals from many others through fine structures called dendrites. The sum of inputs is received by the nucleus. If it receives a sufficiently high signal, it sends a spike of electrical activity. The latter is sent out through the axon. At the end, a structure called a synapse, passes this activity to the next connected neurons. Learning occurs by changing the effectiveness of synapses so that the influence of one neuron on another changes. A simplified representation of a biological neuron is shown in Figure 4.3.

The perceptron. A perceptron is a mathematical model of a biological neuron depicted in Figure 4.3. It takes as input a set X of boolean values.

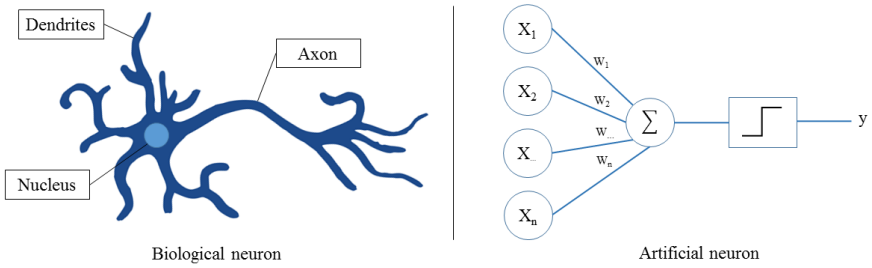


Figure 4.3 – *(left) Representation of a biological neuron. (right) Representation of an artificial neuron.*

The nucleus is modeled by a weighted sum of the inputs $\sum W_i X_i$. The synaptic potential is represented by a hyperbolic tangent:

$$Z = tanh(\sum W_i X_i) \tag{4.1}$$

Finally, the model uses the *Heaviside step* function to binarize the output Y, as follow:

$$Y = \begin{cases} 0 & \text{si} \quad Z < 0 \\ 1 & \text{si} \quad Z \geq 0 \end{cases} \tag{4.2}$$

4.2.3 Why only now?

Why deep learning has only recently become recognized as a trustful, powerful and essential technology as the first experiments with artificial neural networks were conducted in the 1950s. The most noticeable development is that, nowadays, we can provide to algorithms the required resources to succeed: large sets of data and powerful hardware.

Size of datasets. The size of CV datasets has increased dramatically the last few years. This was possible thanks to the digitization of the society. As more and more of our activities take place on the internet, many of our information and actions are recorded including pictures and videos. A deep learning algorithm can take advantage of a huge amount of data and, even, exceed human performance. Recently, the YouTube company made publicly available a large-scale dataset [1] that consists of eight million of YouTube labeled video according to a vocabulary of 4700 visual entities. Beside, the ImageNet project [25] created a large visual dataset designed

for use in visual object recognition which contains over ten million of labeled images.

Size of models. Deep neural networks algorithms are neural networks with higher depth.

> *There is no universally agreed upon threshold of depth dividing shallow learning from deep learning. Most researchers in the field agree that deep learning has more than one nonlinear layers and more than 10 is considered has very deep learning. In Schmidhuber et al. [126].*

Initially, the number of neurons in artificial neural networks were limited by hardware capabilities. Neural networks have been relatively small until quite recently. Today, the number of neurons is mostly a design choice. Some artificial neural networks [19] have nearly as many connections per neuron as a cat ($\approx 10^{13}$).

The explosion of the neural network model sizes is due to faster computers with larger memories. Larger networks are able to achieve higher accuracy on more complex tasks. In 2017, the current hardware built by NVIDIA and dedicated for learning deep neural networks is the *NVIDIA DiGiTS DevBox*. It cost 15,000 dollars and includes the following hardware:

- Four TITAN X GPUs with 12GB of memory per GPU
- 64GB DDR4
- Asus X99-E WS workstation class motherboard with 4-way PCI-E Gen3 x16 support
- Core i7-5930K 6 Core 3.5GHz desktop processor
- Three 3TB SATA 6Gb 3.5" Enterprise Hard Drive in RAID5
- 512GB PCI-E M.2 SSD cache for RAID
- 250GB SATA 6Gb Internal SSD
- 1600W Power Supply Unit from premium suppliers including EVGA

Why the *NVIDIA DiGiTS DevBox* contains four powerful Graphics Processing Units (GPU)? In Section 4.4.3, we present a type of deep neural network – called Convolutional Neural Network – that can take an image as input and apply an analog operation of filtering to it. A filtering operation consists of computing a small function on each pixel of the image. It is a highly parallelizable operation that is handled efficiently by a GPU.

4.3 Technical keys to understand Deep Learning

4.3.1 The multilayer perceptrons

The basic example of a deep learning model is called a feedforward neural network or a multilayer perceptrons (MLP).

The goal of a MLP is to find a mathematical function f that maps a set of input values to outputs. In the recognition process in CV, as explained in Section 4.1, f is the function that will map an input data x to a categorical variable y. In other terms, it aims to learn parameters θ of the function $f(x, \theta)$ that result in the best approximation function $f : x \rightarrow \hat{y}$ of a specific classification problem.

Networks are called so because they contained a chain of many simpler vector-to-vector functions called *layers* which makes possible to write the function $\hat{y} = f(x, \theta)$ in the form $\hat{y} = f_4(f_3(f_2(f_1(x, \theta_1), \theta_2), \theta_3), \theta_4)$. In this case, the function f is composed of four layers as depicted in Figure 4.4. Each layer can be seen as a mathematical function providing a new representation of the input. They are designed to achieve a statistical generalization. The number of layers defines the *depth* of the model. The first one is called the *input layer*, the last one is called the *output layer* and the middle ones are called *hidden layers* as their values are not given by the data.

We note that this network is called feedforward because information flows from the input x to the output y. There are no *feedback* connections in which outputs of a layer are fed back onto itself. When feedback connections exist in a layer, it is called *Recurrent Neural Layer*. These networks are studied in Section 4.4.4.

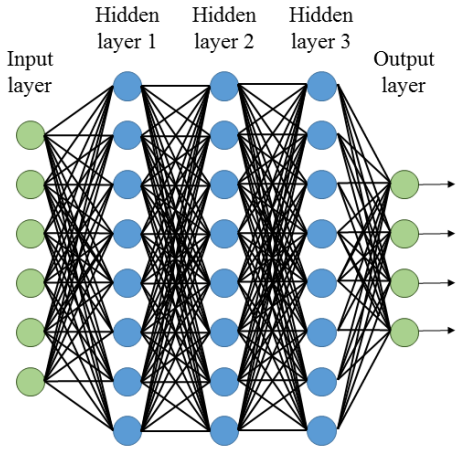


Figure 4.4 – *Schema of a multilayer perceptrons of depth 4*

A layer of a MLP is composed of many units called artificial neurons (AN) that act in parallel. The number of AN in a layer defines its *width*. An AN is an evolution of the perceptron described above and is a vector-to-scalar function. Units are called neurons as they receive inputs from many other previous units and compute their own activation value. An artificial neuron outputs a weighted sum of its inputs followed by an activation function:

$$Z = activation\_function(\sum W_i X_i) \tag{4.3}$$

where X is the input vector, W is the neuron's weights, Z is the output scalar and the operation $\sum W_i X_i$ defines a linear mapping of inputs. An activation function is used here in order to add non-linearity to the transformation. It exists many activation functions but the three most popular in the state-of-the-art are the tangent-hyperbolic, the sigmoid and the rectified linear unit (see in Figure 4.5).

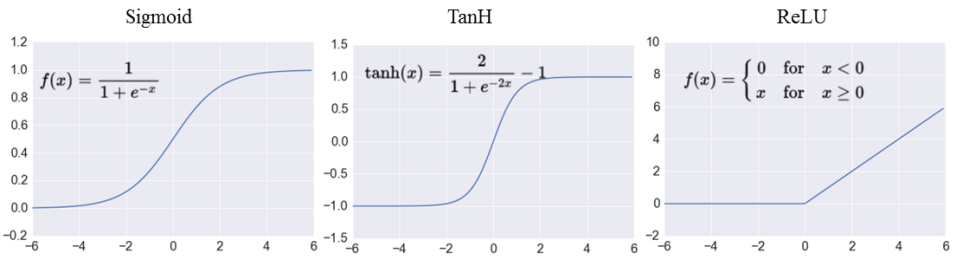


Figure 4.5 – *The three most popular activation functions: (right to left) the sigmoid, the tangent-hyperbolic and the rectified linear unit function.*

Learning algorithms can be described with fairly simple ingredients: a set of data, a cost function, an optimization procedure and a model.

4.3.2 Training a feedforward neural network

The task of a neural network f consists of using a set of labeled data in order to minimize the differences between its output $\hat{y}$ and the label y of a given input x though a *cost function* – also called loss function – and an optimization procedure. To accomplish this task, we *train* the model by updating the parameters θ of f to be the best approximation function $\hat{y} = f(x, \theta)$.

The common optimization procedure to train a network is the *back-propagation* algorithm. Its name comes from the backward propagation of errors procedure [123]. This algorithm works in two steps:

1. **Propagation:** When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired label, using the cost function, and an error value is computed. This error is then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the error.

2. **Weight update:** Backpropagation uses these error values to compute the gradient of the cost function. This gradient is fed to the optimization method which uses it to update the weights in order to minimize the cost function.

The backpropagation step is repeated until the network has processed several times the whole dataset. A complete pass through the whole dataset is called an *epoch*. So, by the end of the first epoch, the model will have been exposed to every sample in the training set once.

If the performance of a neural network for a particular task highly depends on its architecture, several training meta-parameters has to be chosen carefully. We develop a common iterative loop to train a neural network in Algorithm 2.

Algorithm 2: Common iterative loop for training procedure of a neural network

Inputs :

- A training set: $\mathcal{L} = \{(d^i, y^i)\}_{i=1...N}$ where d^i are inputs and y^i are desired labels.
- The neural network model: $\mathbb{M}_\theta$;

Output :

- The trained model: $\mathbb{M}_\theta$.

Parameters:

- The desired number of epoch: E
- The starting learning rate: λ
- The number of data in batches: B
- The cost function: $\mathcal{C}$

Initialize the values of θ ;
$i \leftarrow 0$;
$nbDataSaw \leftarrow 0$;
while $i < E$ **do**
    Extract a subset $(D, Y) = \{(d^i, y^i)\}_{i=1...B}$ from $\mathcal{L}$;
    $nbDataSaw := nbDataSaw + B$;
    `/* Backpropogation procedure */`
    $\hat{Y} = \mathbb{M}_\theta(D)$;
    $\Delta\theta = \frac{\partial \mathcal{C}(\hat{Y}, Y)}{\theta}$;
    $\theta \leftarrow \alpha \times \theta - \lambda \times \Delta\theta$;
    if $nbDataSaw = N$ **then**
        Randomly shuffle $\mathcal{L}$;
        $nbDataSaw \leftarrow 0$;
        $i := i + 1$;

The *Stochastic Gradient Descent* (SGD) algorithm optimizes gradient descent and minimizes the loss function during network training (see lines 9 and 10 of algorithm 2). It is named stochastic as it implies *randomness*. The learning rate λ is used to give a weight to the current update. We often decrease the learning rate while the number of epoch increases to slowly approach a local minima. The parameter B defines the number of training samples that is going to be propagated through the network at each iteration. Using batches in SGD allows to reduce the variance of the gradient updates (applying the average of the gradients in the batch), and accelerate the model's optimization.

For more information about existing layers, optimization procedures, applications and on Deep and Machine Learning technology in general, we refer the reader to the book *Deep Learning* [41] written by Ian Goodfellow, Yoshua Bengio and Aaron Courville, freely readable at `http://www.deeplearningbook.org`.

To summarize, a deep learning model is a chain of simpler functions called *layers*. It exist many different layers in order to handle different data, such as images, vectors and different challenges, such as handling sequences, etc.

Deep neural networks are modeled according to:

1. The nature of the data;
2. The nature of the output;
3. The nature of the problem;
4. The hardware environment.

Layers can also be tuned by their width and their activation function. Among many others, theses parameters are called *hyper parameters* or yet *meta parameters* as they cannot be learn directly from the data.

4.4 Technical details of deep learning elements

As explained above, the design of a neural network model is mostly driven by the problem to solve. In this section, we explain in details the essen-

tial elements for a deep learning recognition process. We present several essential elements of deep learning for hand gesture recognition:

- As we study a classification task, we introduce the softmax activation function which is useful for outputting a class-conditional probability vector, essential to represent categorical variable, and the cross-entropy cost function.
- Generally in CV, inputs are often images. We present the convolutional layer which is a layer specialized in processing grid shape inputs.
- To take into account the dynamic aspect of videos, we introduce Recurrent Neural Networks (RNN) which are designed to handle sequence of data.
- As 3D datasets are time consuming, hard to capture and using deep learning algorithm with datasets of small size leads to a bad generalization called *overfitting* during the training phase. A way to fix this problem is to use a *transfer learning* strategy using an other similar larger dataset to extract features.

4.4.1 Softmax function

The output of a classification task is a categorical variable (in opposition with quantitative variable). By definition, a categorical variable has a fixed number of possible and discrete value. For example, in an animal recognition application based on images, the categories could be: fish, bird, cat and dog. Each sample in the dataset is assigned to one of those finite categories. They differ from quantitative variables in that the distances from one category to another are equals, regardless of the number of categories.

We could use a single scalar to represent the outputs but the distances between each category would not be equal. In order to fix this issue, deep learning models designed for classification use the one-hot encoding to represent their outputs. It consists of a vector which the size is equal to the number of categories, fill with 0 and a 1 in the cell of the category to which the input belongs. We can also see this encoding scheme as a

particular stochastic vector which represents the probability that an input belongs to each category, called a class-conditional probability vector.

In order to output a stochastic vector, a model needs two elements. First and obviously, the last layer of the model needs to have the size of the number of categories. Second, to compute a class-conditional probability vector, the last layer uses the *softmax* activation function. The output of the softmax function is a categorical probability distribution which indicates the probability that the input belongs to any of the classes.

Let K be the number of categories in a classification task and Z the weighted sum of the input of the last layer (see in Section 4.3.1). The softmax function is defined as follows:

$$softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4.4}$$

where $j = 1 \dots K$ and $\mathbf{z}$ is the output of the last layer. The second purpose of this function is, using the exponential terms, to highlight largest inputs and to suppress all significantly smaller ones.

4.4.2 Cross-entropy cost function

A primordial aspect of a deep neural network training is the choice of the *cost function*. Cost functions, for neural networks, are more or less the same as for any trainable classifiers. We use the *cross-entropy* between ground-truth and model's predictions as the cost function.

Let $\mathcal{L} = \{(\beta^i, y^i)\}_{i=1\dots N}$ be a set of labeled data. The class-conditional probability output of the deep neural network is noted $\hat{y}^i = f(\beta^i, \delta)$. The cross-entropy expression is given by:

$$L(W) \;=\; -\frac{1}{N}\sum_{i=1}^{N}\left[y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)\right] \tag{4.5}$$

The lowest is $L(W)$, the best is the quality of the approximation function f. Figure 4.6 shows the values of the cross-entropy error according to values of y and $\hat{y}$.

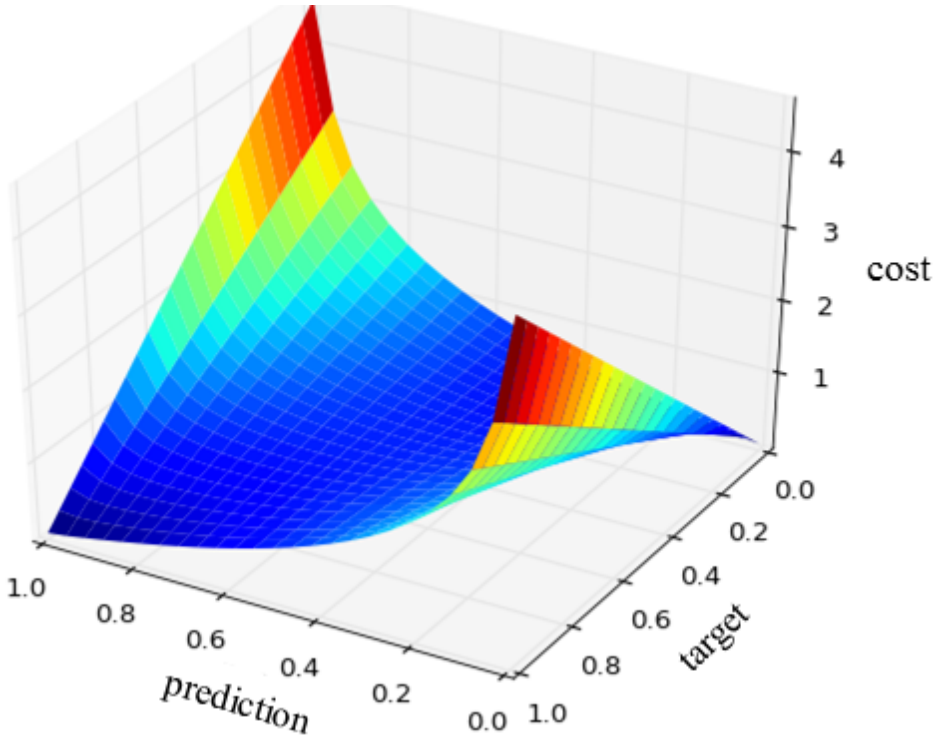


Figure 4.6 – *Values of the cross entropy cost function $L(W)$ according to values of y (target) and the output of a classifier $\hat{y}$ (prediction). Image reproduced from* `https://github.com/matplotlib/matplotlib/issues/6027`.

4.4.3 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a specialized kind of neural network for processing data that has a grid-like topology. It includes time-series of vectors, which are grid-like data when concatenated, and, of course, images which are 2D grids of pixels. In 1998, LeCun *et al.* [69] proposed a pioneering 7-level CNN called LeNet-5 designed to classify 32×32 digits images extracted from bank checks. As the raw input of hand gesture recognition algorithms are generally 2D images, we introduce motivations of a CNN design.

A) Motivations

In traditional neural network layers such as multilayer perceptrons, introduced in Section 4.3, every output interacts with every input. It means that the number of parameter of a neural network model is proportional to its input size. In addition, if they are m inputs and n outputs, the matrix multiplication requires $m \times n$ parameters and the algorithm have

a $O(m \times n)$ runtime. When processing an image, the input might have thousands or millions values and a conventional multilayer perceptrons will see its number of parameter and runtime explode. Also, such network architecture does not take into account the spatial structure of the image. By handling images as vector of pixels, it does not allow the network to benefit of the strong spatially local correlation present in images which are important features in recognition task.

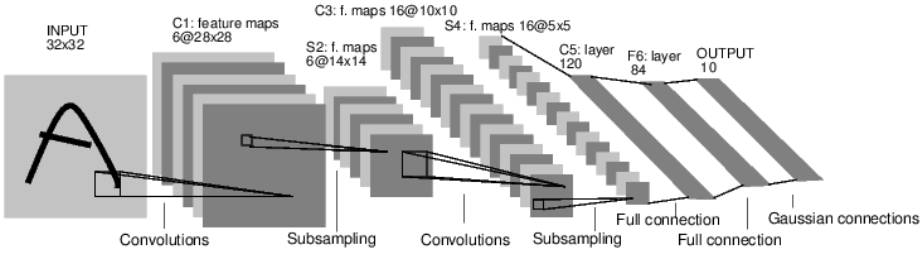
B) A biologically inspired version

The design of the CNN have been guided by neuroscience. The CNN history begins with neuroscientific experiments long before the first one has been developed. Neurophysiologists David Hubel and Torsten Wiesel have collaborated for several years to determine many of most basic facts we know about how the mammalian vision system works [55]. They observed how neurons in a cat's brain responded to images projected in precise locations on a screen. They identified two basic visual cell types. The *simple cells* in the early visual system respond to very specific patterns of light, such as oriented bars, but respond hardly to complex patterns. Beside, *complexer cells*, with a larger receptive fields, are invariant to small shifts in feature positions.

C) Design

To simulate the visual cortex, a CNN architecture is composed of stacked bunch of distinct layers. An example of a CNN architecture is given in Figure 4.7.

First, there is the convolutional layer which is the core of a CNN. The layer's parameters consist of a set of learnable filters which have a small size but slide over the whole image. A filter is convolved across the width and the height of the grid-like input. It follows an activation function which produces a 2-dimensional response map, one for each filter. As a result, the network learns filters that activate when they detect some specific features at some spatial positions in the input. A simple convolution step is depicted in Figure 4.8.



- Layer C1 is a convolution layer with 6 response maps of 28×28 filters.
- Layer S2 is a subsampling layer with 6 response maps of 14×14 filters.
- Layer C3 is a convolution layer with 16 response maps of 10×10 filters.
- Layer S4 is a subsampling layer with 16 response maps of 5×5 filters.
- Layer C5 is a multilayer perceptron also called fully connected layer of size 120.
- Layer F6 is a fully connected layer of size 84.

Figure 4.7 – *Architecture of LeNet-5 by Lecun et al. [68], a Convolution Neural network designed for digits recognition. Image reproduced from [69].*

The convolutional layer can be represented by a function $\mathcal{C} : \mathbb{R}^{h \times w \times c} \mapsto \mathbb{R}^{h \times w \times n}$ where h, w and c are respectively the height, the width and the number of channel of the input grid and n the number of filters learned by the layer. The convolutional layer is designed to emulate properties of simple cells described above as it tries to learn simple and local features in the input grid.

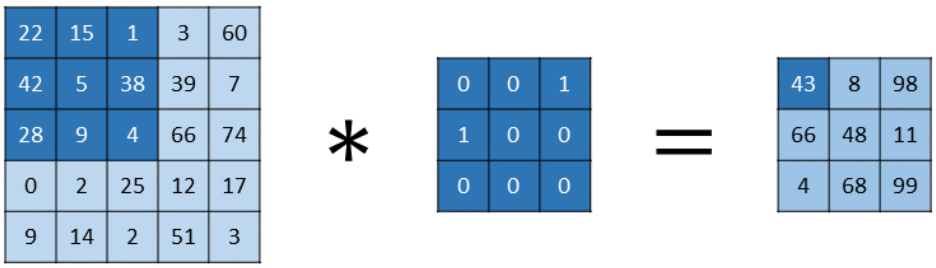


Figure 4.8 – *A simple illustration of two dimensional convolution operation.*

The convolutional layer is followed by a subsampling layer performed by a non-linear operation called *pooling*. If there are several non-linear functions to implement the pooling, *max pooling* is the most common. It partitions the input image into a set of non-overlapping region and outputs maximums. A simplified scheme of a max pooling layer is given in Figure 4.9. The intuition is that exact feature positions are less important than their locations relative to other features. The pooling layer serves to progressively reduce the size of the representation, the number of parameters and the amount of computation while the information flow through the network. It provides also a form of translation invariance. The pooling layer is inspired by complex cells as it allows the network to be invariant to small shifts of the feature positions.

The pooling layer can be represented as a function $\mathcal{P} : \mathbb{R}^{h \times w \times c} \mapsto \mathbb{R}^{h/p_1 \times w/p_2 \times c/p_3}$ where h, w and c are respectively the height, the width and the number of channel of the input grid and p_1, p_2, p_3 are fixed hyper-parameters of the pooling layer.

D) Conclusion

The basic strategy of convolutional feature detection followed by pooling is repeatedly applied as we move deeper into the network. It allows the

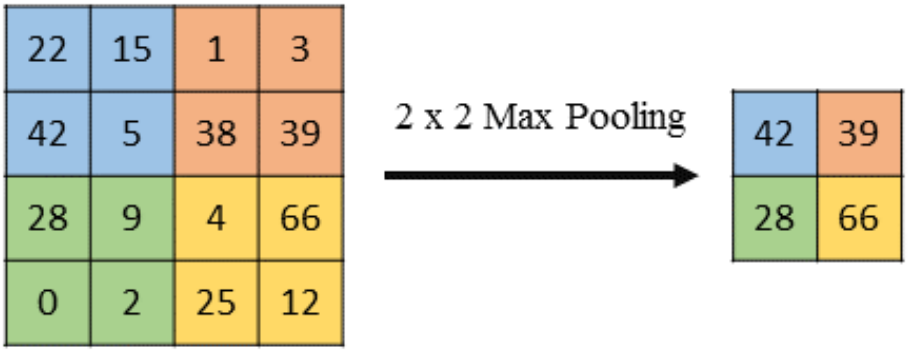


Figure 4.9 – *A simplified scheme of a* 2×2 *max pooling layer.*

CNN to learn from low to more abstract features. Stacking many layers leads to non-linear local filters that become more and more global. This concept is illustrated in Figure 4.10.

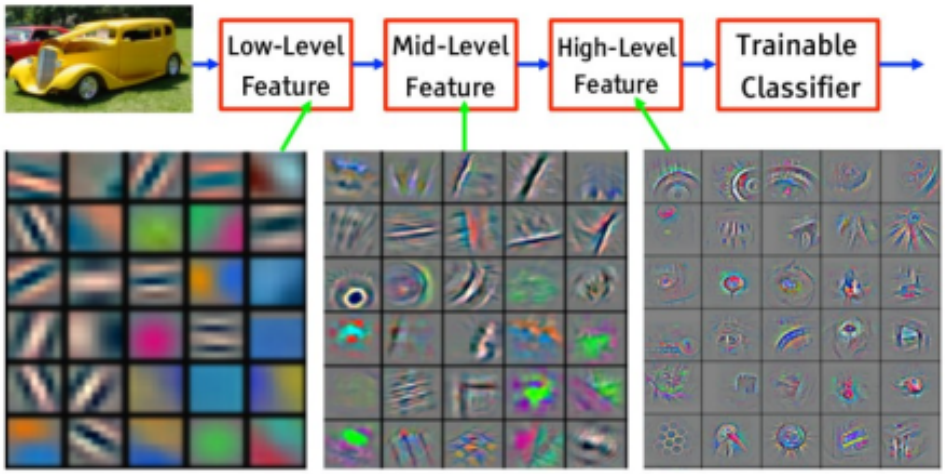


Figure 4.10 – *An illustration of how stacked convolutionnal layers learn from low to high level features in images, e.g. pixels to edges to motifs to parts of objects. Image reproduced from [10].*

Generally, the output of stacked convolutional layers are finally flattened in order to use learned features as inputs to other types of layers that need vector as input. All together, they form what we call a Convolutional Neural Network architecture that allows to perform some classification tasks on images.

This independence from prior knowledge and human effort in feature design is a major advantage. The network is able to learn filters that in traditional algorithms were hand-engineered. In addition, each filter is used across the entire image. It means that all neurons in a given convo-

lutional layer respond to the same inputs. This property, so called *weight sharing*, reduces the number of learned parameters, lowering the memory requirements for training and running the network.

In addition to images, CNN can handle 1-D temporal sequences represented as stacked vectors. The idea behind 1-D convolutions of temporal sequences is to share parameters across time. The output of a sequence convolution is a sequence where each vector of the output is a function of a small number of neighboring vector of the input.

4.4.4 Recurrent Neural Networks

Many CV recognition applications need to handle dynamic data, as the term *dynamic* referred to, which take temporal sequences as input. Recurrent neural networks (RNN) are a family of neural networks, introduced by Rumelhart *et al.* in 1985 [123], for handling sequential data as input. As a CNN is specialized for processing grid shaped inputs, a recurrent neural network is a neural network specialized for processing vector sequences.

A) Motivations

Compared to CNN, most recurrent networks can process sequences of variable length. Recurrent networks share parameters in a different way and each output can be a function of the whole previous vectors of the input. For the simplicity of explanation, we refer to RNNs as operating on a sequence that contains n vectors $v(t)$ with $t = 1 \dots n$. In practice, recurrent networks usually operate on sequences with different lengths.

B) Time-series representation

Unlike feedforward neural networks, RNNs use an internal memory to process arbitrary sequences and a relation between their output and their input. We consider the dynamical system: $s(t) = f(s(t-1), \theta)$ where $s(t)$ is the current state of the system and θ are the transition parameters. This equation is recurrent because the state of the system at the time t is dependent of the system's state at time $t-1$.

In the case of RNN, we rewrite the function s to add an input vector h exterior to the system: $y(t) = s(h(t-1), x(t), \theta)$ where h is the internal memory (or the internal state) of the layer. We can train the parameters θ such as the output $y(t)$ extract features which will be a representation of the sequence from the time step 1 to t. This lossy representation might select some aspects of the past sequence while forgetting irrelevant ones. For example, in order to recognize only swipe dynamic hand gestures, the RNN will probably not store information when the hand is not moving.

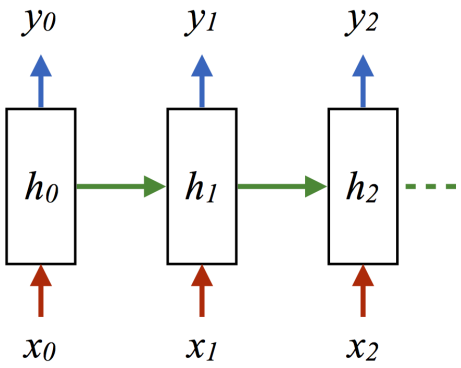


Figure 4.11 – *A scheme of a simple RNN also called Elman network [32].*

A simple RNN layer is depicted in Figure 4.11. The figure does not specify the activation function for the hidden units. Assuming the hyperbolic tangent activation function is used, the equation of the RNN layer is:

$$h(t) = tanh(W_1 x(t) + W_2 h(t-1)) \tag{4.6}$$

$$y(t) = tanh(W_3 h(t)) \tag{4.7}$$

where W_i are weight matrices. This is an example of a recurrent network that maps an input sequence to an output sequence of the same length.

C) The challenge of long term dependencies

Like most neural networks, recurrents are old. In the beginning of the 1990s, the vanishing gradient problem emerged as an obstacle to RNN's

performance. During training stage, gradients propagated over many stages tend to vanish. This problem is particular to recurrent networks. Multiplying a weight w by itself many times result in a power function which either vanish or explode following if w is inferior at 1 or not. Also, as the input sequences become larger, simple RNNs are unable to learn long term dependencies. In 2017, most effective sequence based neural networks used in practical applications are called Gated-RNNs. These include the Long Short-Term Memory (LSTM) layer introduced by Hochreiter and Schmidhuber in 1997 [52].

LSTMs are designed to handle long-term dependencies and to fix the vanishing gradient problem. They have proven their usefulness on a large variety of problems and are now widely used. The amazing idea behind the LSTM is the addition of three gates that handle different mechanisms:

- The gate f_t handles a *forgetting mechanism* to decide which information of the current state is now useless and should be forgotten.
- The gate i_t handles a *saving mechanism* to decide which information of the current input should be saved.
- The gate o_t handles a *focus mechanism* to decide which information of the state is useful for the current step.

Thus, where a RNN changes its memory cell at each time step in an uncontrolled way, a LSTM transforms its memory in a very precise way. It uses specific learning mechanisms to choose which pieces of information to remember, to update, and which to pay attention to for the current time step as depicted in Figure 4.12.

Let us now take a look at the mathematical formulation of the LSTM, which consists of five equations. Let x_t be the input vector, y_t the output vector, h_t the cell state vector, W and U the learned parameters, σ_g the sigmoid activation function and $\circ$ denote the Hadamard product (element-wise product). The first three equations are for the new mechanisms described above:

$$f_t = \sigma_g(W_f x_t + U_f y_{t-1}) \tag{4.8}$$

$$i_t = \sigma_g(W_i x_t + U_i y_{t-1}) \tag{4.9}$$

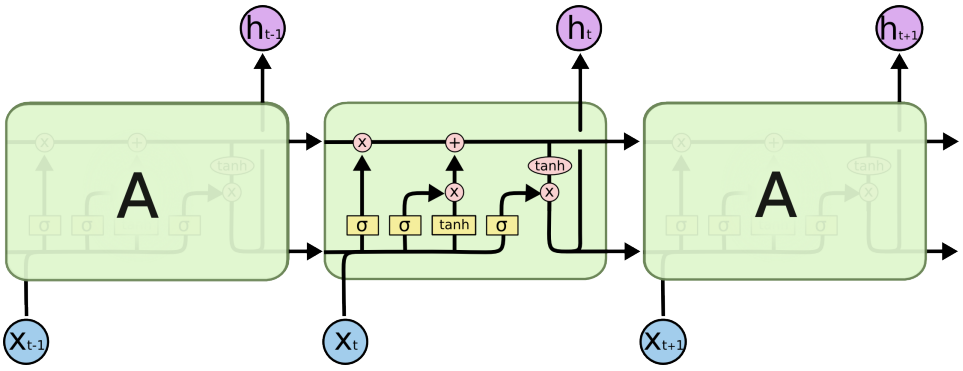


Figure 4.12 – *A simplified Scheme of a LSTM layer. Image reproduced from* `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

$$o_t = \sigma_g(W_o x_t + U_o y_{t-1}) \tag{4.10}$$

We can see that f_t, i_t and o_t take their decisions based of the previous output of the LSTM layer and its current input but do no share their parameters. The next equation has for purpose to update the cell state:

$$h_t = f_t \circ h_{t-1} + i_t \circ tanh(W_c x_t + U_c y_{t-1}) \tag{4.11}$$

The first part of the equation $f_t \circ h_{t-1}$ uses the f_t gate in order to remove useless information from h_{t-1}. The second one uses i_t and new parameters to choose the relevant information to save from x_t in order to update the cell state. Finally, the LSTM computes its output using its cell state h_t and chooses which current information is relevant using the *focus* gate o_t:

$$y_t = o_t \circ tanh(h_t) \tag{4.12}$$

D) Overfitting problem and transfer learning

The goal in a classification process is to propose a classifier which works correctly on new previously unseen inputs. The ability to handle unobserved inputs is called *generalization*.

Typically, a dataset is composed of two non-overlapping sets. The first, called *training set*, is composed of data on which the algorithm is learning from. The second, called *test set* is composed of data unseen by the algorithm during the training phase. The classifier has to minimize error measures between its outputs and ground-truths through an optimization

procedure. This error measure is called *training error* when computed on the training set and *generalization* or *test error* when computed on the test set. What determines the effectiveness of a learning algorithm is its ability to make the training error small and to reduce the difference between the training and the test error called *generalization gap*.

The *capacity* of a deep learning model is its ability to fit a particular problem. Two main hyper-parameters define the capacity of a model; its *depth* and its *width*. A model with a low capacity may be unable to fit the training set. A model with a high capacity may overfit by learning specific properties of the training set that do not serve for the generalization. A model with a high capacity can solve complex tasks but it needs more data to avoid the overfitting. To resume, harder is the task, higher has to be the depth and the width of the model and, consequently, the amount of data needed increased. Figure 4.13 shows relationships between the capacity of a model and error measures. Unfortunately, we have no chance to find the best network architecture that generalizes the training set perfectly as unlimited different solutions exist.

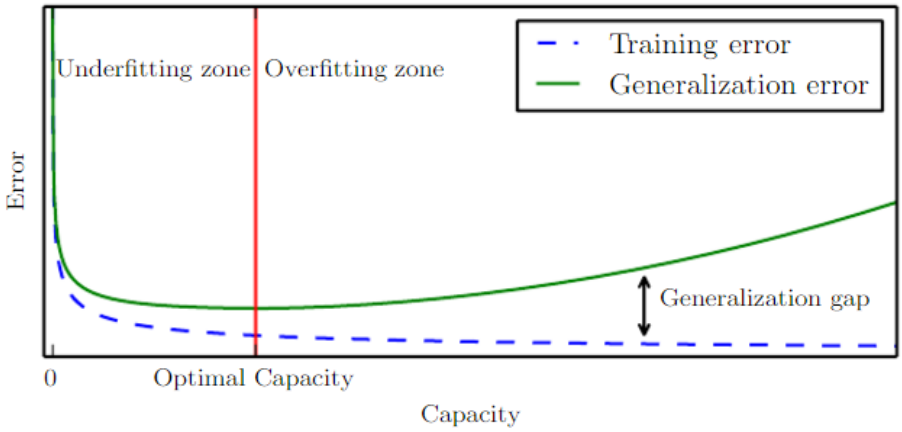


Figure 4.13 – *Typical correlation between the capacity of a model and error measures. At the left of the optimal capacity, we could increase the capacity to find a better generalization of the training set. This state is called underfitting. To the right of the optimal capacity, the model is too large or the training set is too small and the algorithm starts to learn training data specification. It results of a decreasing training error, an unfortunate increasing of the test error and a bigger generalization gap. This state is called overfitting. Image reproduced from [41].*

For image classification or object detection challenges, the CV community have access to very large datasets, such as the Open Images dataset [65] which is composed of 10,000,000 labeled images. In the field of

hand gesture recognition, datasets are composed of only thousands of sequences. If creating more data would be the best way to avoid overfitting, it is time consuming and not always possible. Nevertheless, it exists methods and tricks to prevent the model to overfit, here are some:

- Use a smaller model;
- Use weight decay. Weight decay is a regularization term added to the cost function that penalizes big weights. When the weight decay coefficient is big, the penalty for big weights is also big, when it is small weights can freely grow.
- Use a dropout strategy. Dropout randomly makes nodes in the neural network "drop out" by setting them to zero, which push the network to rely on other features. It results in a more generalized representation of the data;
- Use data augmentation. As deep networks need large amount of training data to achieve good performance, we can artificially create additional training data. For example, to train a CNN architecture, new images can be created through random rotations, shifts, etc;
- Use early stopping. Stop the training phase before the model starts to learn training set specifications;
- Use transfer learning.

Focus on transfer learning. First of all, we give definitions of a *domain* and a *task* as defined by Pan *et al.* [108] in their survey on Transfer Learning. A *domain* $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ where $X = \{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$. Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a *task* consists of two components: a label space $\mathcal{Y}$ and an objective predictive function f (denoted by $\mathcal{T} = \{\mathcal{Y}, f\}$), which is not observed but learned from the training data which consists of pairs $\{x_i, y_i\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. For example, in the field of hand pose estimation, $\mathcal{X}$ is the image depth space, x_i are hand depth images, $\mathcal{Y}$ is $\mathbb{R}^{3*k}$ where k is the number of joints in the hand model, y_i are 3D joint

positions for each samples in the dataset and, finally, f is the mapping regression function defined as $f : \mathcal{X} \mapsto \mathcal{Y}$ learned from the training set.

We define a source domain $\mathcal{D}_S$ and a target domain $\mathcal{D}_T$. More specifically, we denote $\mathcal{D}_S = \{(x_{1_S}, y_{1_S}), \ldots, (x_{n_S}, y_{n_S})\}$, where $x_{i_S} \in \mathcal{X}_S$ is a data sample and $y_{i_S} \in \mathcal{Y}_S$ is the corresponding label. Similarly, $\mathcal{D}_T = \{(x_{1_T}, y_{1_T}), \ldots, (x_{n_T}, y_{n_T})\}$. Note that, in most cases, $0 \leq n_T \ll n_S$.

Given a source domain $\mathcal{D}_S$ and a learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a learning task $\mathcal{T}_T$, transfer learning aims to help the learning of the target predictive function f_T in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$ but similar.

Learning features on images is a complex task. Beside, CNN architecture contains lot of parameters and, so, are not usually trained from scratch with random initialization. This is because it is relatively rare to have a *target* dataset of sufficient size to train a network with a depth large enough to handle the complexity of the task. Instead, it is common to train a CNN on an other larger *source* dataset and then use the, so called, *pre-trained* weights either as an initialization or a fixed feature extractor for the task. *Transfer learning* and depends on two major factors: the size of the *source* dataset and its similarity to the original dataset.

4.5 Conclusion

In this chapter, we studied why and how researchers in the field of Computer Vision migrate from handcrafted to learned-based features algorithms, so called deep learning or also deep neural networks. The main difference between them is that handcrafted features use human ingenuity and problem-specific knowledge to extract relevant features from the data while deep learning algorithms learn them from the data. Last years, deep neural networks have proven their wonderful effectiveness of many area of research. Nevertheless, both of these approaches have their strengths, their weaknesses and their scope of application.

Deep learning became a hot topic since 2010 thanks to the digitization of the society that exponentially increased the size of dataset. Also, new

and powerfull hardware capacities allow researcher to use much deeper neural networks. Yann LeCun is a French computer scientist with many contributions in machine learning and in computer vision. He is also a founding father of convolutional neural networks. The number of its paper's citations through time in Figure 4.14 proves the trend of deep learning in the computer vision community.

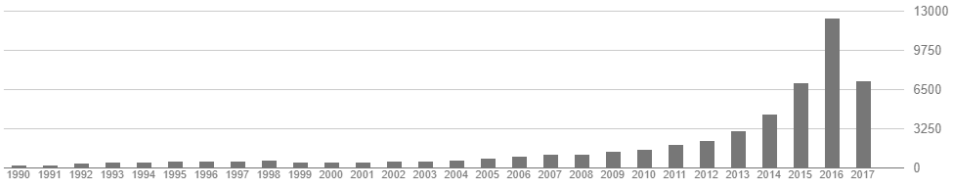


Figure 4.14 – *Number of paper's citation of Yann Lecun, french computer scientist and founding father of convolutional neural network, from 1990 to July 2017.*

One of the drawbacks of using a deep neural network is that its hyper-parameterization is conducted by human intuitions and experimentations. Unfortunately, to the best author's knowledge, there is no mathematical way to choose the design of a neural network according to the challenge and the data. Researchers have no chance of finding the optimal solution that will perfectly fit their problematic. However in 2017, Kaiser *et al.* [58] from the Google company, introduce a single model that can fit a variety of tasks including translation, language parsing, speech recognition, image recognition, and object detection in their paper *One Model To Learn Them All*. It opens the door to a future where, maybe, the hyper-parameterization of a model is not be a problem anymore.

In the case of the use of deep learning for hand gesture recognition task, the main negative factor is the size of current available dataset. Learning a deep neural network using a small amount of data leads to an overfitting of the model. Extracting a very large amount of no naturally digitized 3D data is time consuming and not always possible. There are still methods to learn less deeper network with less data as smart transfer learning strategy and data augmentation.

5 Online detection and recognition of hand gestures using a deep learning approach

Contents

5.1 Introduction

Deep neural networks have proven their effectiveness in various challenges, improving recognition rates substantially for various image classification tasks. Furthermore, motivated by their success for images and videos, many research works have appeared very recently proposing models, such as convolutional neural network, for learning hand pose features.

Convinced of the usefulness of the pose features to describe hand gestures and motivated by the success of these approaches, we extend the study to online dynamic hand gestures taking over the whole pipeline of the recognition process, from hand pose estimation to the classification step, using deep learning.

We aim to structure such a framework following two statements made in Chapter 3:

1. Hand postures along the sequence are relevant features to describe the gesture.

2. Hand gestures can be efficiently described by the temporal variation of both, hand shape and its motions.

Based on the previous statements, we present in this chapter a new framework based on deep learning for online dynamic hand gesture recognition using a transfer learning strategy. So as to face the main challenges, we propose to revisit the feature pipeline by combining the merits of geometric shape features and dynamic appearance, both extracted from a CNN trained for hand pose estimation problem.

Despite an increasing amount of methods proposed over the last few years, defining an online dynamic hand gesture recognition system robust enough to work in real world applications is still very challenging.

5.1.1 Challenges

In addition to challenges defined in Chapter 3 for dynamic hand gesture recognition, like *inter-class* similarities, *intra-class* dissimilarities and issues linked to the intrinsic properties of the hand, others appear for an online recognition in real time using learned features.

It is hard to include prior specific knowledge in a data-driven learning algorithm. Dynamic hand gestures can be defined by shape variations of the hand during sequences (e.g. fine gestures performed by fingers), or by hand movements (e.g. swipe gestures), but often both. These multiple characteristics, which have to be taken into account, make harder the process of features learning as it have to learn both *spatial* and *temporal* information.

In order to fully extract relevant features of complex hand gestures using raw data, models of neural network need a large number of layers which increase their time complexity. However, the time complexity has to be small enough so that the algorithm can predict a new incoming gesture in real time. Some methods present acceptable runtime results using very deep networks but use a powerful hardware with several GPUs. Currently, this hardware configuration is too expansive and usually unavailable and, so, not suitable for real applications.

We must not forget that, the user in front of the camera is not always performing a gesture. Even worse, the user performs also movements that are not relevant and belong to none of the gesture categories; for example, to come back to a restful position between relevant gestures. Thus, an online gesture recognition task implies a gesture *detection*, also called gesture *localization*.

In addition, in some field of computer vision, researchers have access to millions of data allowing them to train very deep neural networks from scratch. For example, in the field of image classification, Deng *et al.* [25] make publicly available the ImageNet dataset which contains 10 million images. However, it is not the case while working with 3D data making difficult the use of data-driven approaches easily. To our knowledge, the largest dynamic hand gesture dataset contains only $2,800$ gesture sequences [23].

5.1.2 Overview of the proposed framework

To face challenges stated above, we introduce a framework for online hand gesture detection and recognition. It is based on temporal hand shape

and posture learned features. Those data are both extracted from a CNN trained for hand pose estimation. The overview of our framework is depicted in Figure 5.1.

First, we aim to use a transfer learning strategy to extract hand shape and posture features for hand gesture recognition purpose. To do so, we train a CNN for hand pose estimation using the ICVL dataset [144]. Once the training of the CNN is over, we can use it to output two distinct representation for each time step of a hand depth image sequence: hand posture features, noted J_t, which represent hand joints locations and a hand shape feature vector X_t laying in a high dimensional space. We note that if a hand posture sequence is represented by the absolute position of the hand joints, the descriptor X focuses on describing the hand shape.

Thus, original hand depth image sequences, $s_{original} = \{I_t\}_{t=1...N}$, are transformed into two different sets of sequences as follows: $s_{posture} = \{J_t\}_{t=1...N}$ and $s_{shape} = \{X_t\}_{t=1...N}$ for a sequence of N frames.

We fed both sequences in parallel to the $RNN_{skeleton}$ and the RNN_{shape}. Both recurrent networks are made of two stacked LSTM layers and a final fully connected layer. We aim to extract temporal features, respectively, about hand motions and hand shape variations. Finally, we fuse the results of the two networks using a joint fine tuning strategy in order to get a single class-conditional probabilities vector for each time step. The maximum probability is then chosen as the predicted class. We take the predicted class of the last frame as the predicted label of the sequence. Figure 5.1 summarizes the proposed solution.

5.1.3 Motivations

Main considerations that motivated our approach are:

- Data-driven approaches show outstanding results in many recognition tasks.
- Hand posture data are efficient feature for hand gesture description.
- A hand gesture is efficiently described by the temporal aspect of hand shapes and its motions.

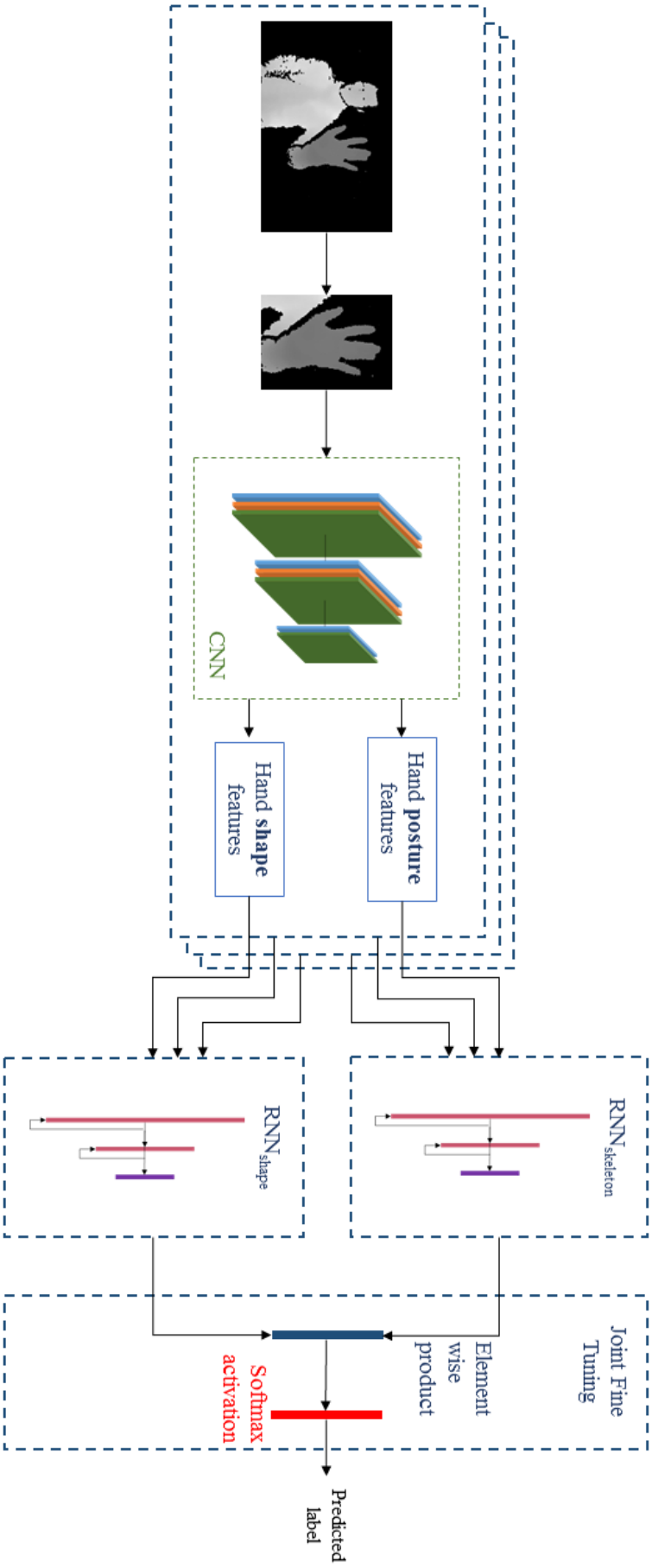


Figure 5.1 – *Overview of our framework. First, two frame-wise features are extracted from hand depth images using a CNN trained for hand pose estimation: hand posture features and hand shape features. Both descriptor sequences are fed to two distinct recurrent networks, called $RNN_{skeleton}$ and RNN_{shape}, in order to extract temporal features, respectively, about hand motions and hand shape variations. Finally, output networks are fused using joint fine tuning and a softmax layer output a class-conditional probabilities vector for each time step. The predicted label of the last frame is use for sequence classification.*

- The transfer learning strategy allows the use of more labeled data than provided in dynamic hand gesture datasets.
- Online hand gesture recognition needs real time processing for real world applications.
- Online hand gesture recognition implies a prior hand gesture detection.

5.2 Deep extraction of hand posture and shape features

In Chapter 3, we highlighted the potential of hand skeletal features in the hand gesture recognition task.

Hand pose estimation is a growing field of research. They focus on creating new algorithms able to retrieve a set of 3D hand joints, called later hand skeletal data, from 2D and/or 3D hand images. The needs of precise mid-air HCI for emerging applications, such as the interaction with a virtual or augmented reality world, attracted much attention in the Computer Vision community [102, 101, 40, 75, 76, 140].

In this section, we introduce a Convolutional Neural Network (CNN) learned to extract hand shape and posture features inspired by Oberwerger *et al.* [101] approach of hand pose estimation. The later uses a two steps method. First, a CNN output an estimated hand pose. Second, they train one CNN for each joint to correct small mistakes in the previous hand pose estimated. As the second step is a computationally expensive and that we do not need millimeter precision for hand gesture recognition, we use in our framework only the CNN of the first step.

5.2.1 Formulation of hand pose estimation problem

The hand pose estimation is formulated as the estimation a set J of 3D hand joints coordinates $J = \{j_i\}$ where $i = 1 \dots K$ and $j_i = (x_i, y_i, z_i)$ from a depth image I as depicted in Figure 5.2. Thus, the goal of hand pose estimation is to find parameters θ of the mapping regression function f

such as $f(\theta) : I \rightarrow J$. Several algorithms of hand pose estimation have been presented in Chapter 2.

Figure 5.2 – *A hand depth image and its K 3D joints. Here, K equal to* 16*: 3 joints by fingers and one for the palm.*

5.2.2 Hand pose estimation dataset

A learned-based hand pose estimation algorithm needs a large amount of hand depth images labeled with their hand skeletal annotation to be trained. There are only one hand gesture datasets which provide both depth and skeletal data. So far and to our knowledge, only De Smedt *et al.* [23] made publicly available such dataset. Moreover, hand skeletal features from this dataset are extracted by the *Intel Real Sense* camera. Consequently, we can not used this dataset for hand pose estimation as the skeletal annotations are estimated from an algorithm and can not be considered as reliable ground-truths.

There are several hand pose datasets in the literrature. The two widely used in the 3D hand pose estimation area are the NYU dataset [147] and the ICVL dataset [144], respectively, captured using a first version of *Microsoft Kinect* and an *Intel Creative* depth sensors.

These devices have two main differences: *Microsoft Kinect* is *a structured light-based sensor* and a long-range camera. The *Intel Creative* use the time-of-flight technology and is a mid-range camera, thus providing more precise and less noisy data. For our experiments, we choose the ICVL dataset, as it contains a larger amount and cleaner data.

5.2.3 Pre-processing step

For a depth image noted I, we first estimate the region-of-interest (ROI) of the hand. It is done assuming the hand is the closest object to the camera, similarly to Tang *et al.* [144]. Second, we refine this estimation using a 3D bounding box of size $128 \times 128 \times 300$. Finally, we compute and save the 3D center of mass P^{com} of points laying inside of the cube. We crop the depth image around the 3D bounding box and normalize the patch values to $[-1, 1]$. Outlier points that are outside of the box depth are assigned a value of 1. The resulting image is noted I^*.

On the annotated skeleton J, we remove the global hand position in the scene by substracting the P^{com} coordinates to skeleton joints as follows:

$$J^* = \{j_i - P^{com}\}_{i=1...K} \tag{5.1}$$

where j_i is the i^{th} joint in J, P^{com} is the center of mass of the ROI of the hand and J^* is the skeleton centered around P^{com}.

The pre-processing step is depicted in Figure 5.3.

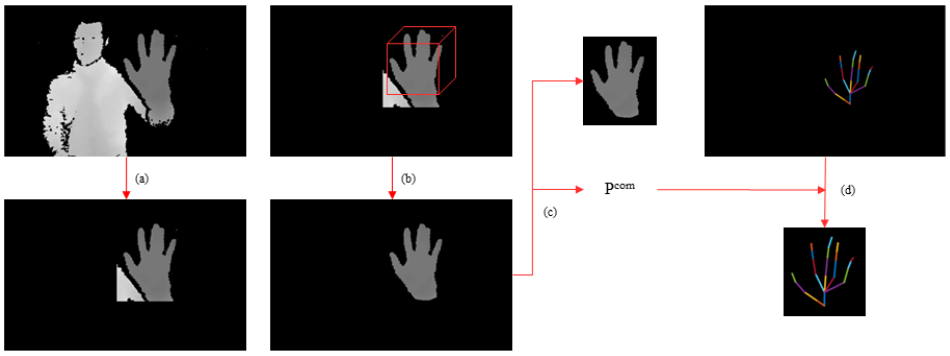


Figure 5.3 – *The pre-processing step for hand pose estimation. (a) First, we estimate the region-of-interest (ROI) of the hand assuming the hand is the closest object to the camera. (b) Second, we refine the estimation using a 3D bounding box around the center of the mass. (c) From this, we can extract a cropped image of the hand and we compute its center of the mass P^{com} in the original image space. (d) Using P^{com}, we remove the global hand position from the skeletal ground-truth in the scene by substracting the P^{com} coordinates to each skeleton joints.*

5.2.4 Network model for predicting 3D joints locations

Inspired by Oberweger *et al.* [101], we consider the hand pose estimation algorithm based on a CNN architecture using prior enforcement. The idea

comes from that, given the physical constraints over the hand, there are strong correlations between 3D joint locations. Wu *et al.* [163] showed that a lower dimensional space that $3 \times K$ is sufficient to parameterizes a 3D hand pose of K joints. The network, further called CNN_{hand_pose}, architecture is depicted in Figure 5.4. The implementation of the prior enforcement is made by introducing a *bottleneck* in the penultimate layer, having a smaller size than the final one which outputs the 3D joint coordinates. The linear mapping between the lower dimensional space and the final output is kept by not adding any activation function to the bottleneck layer.

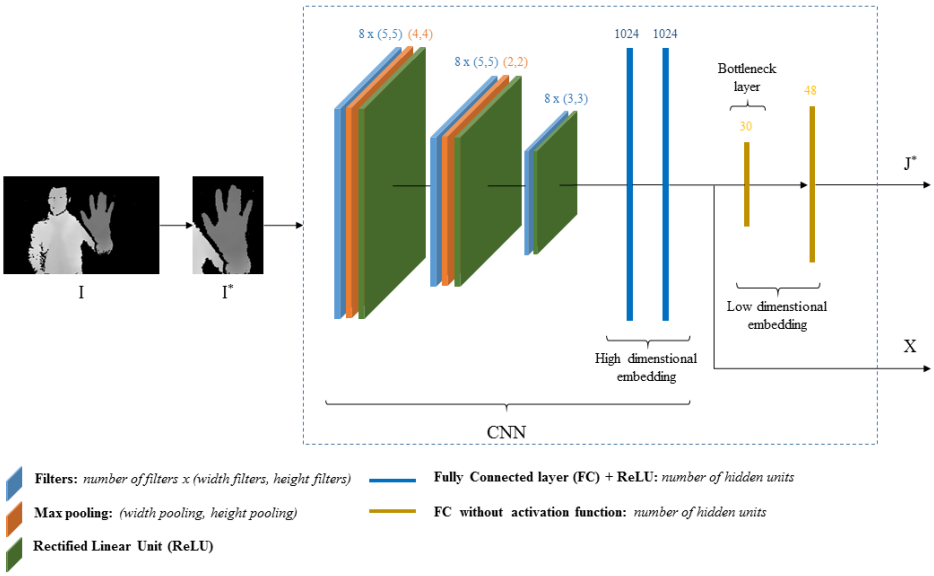


Figure 5.4 – *Architecture of the CNN for hand shape and posture extraction using prior enforcement. The model takes as input a cropped depth image I^* around the hand. It use a CNN architecture to map this image to a high dimensional space vector X, laying in $\mathbb{R}^{1024}$. The latter contains large, global and local information about the hand shape. Follows a "bottleneck" layer with a smaller size that the desired output to model the physical constraints over the hand topology. The network finally output hand skeletal features J^* centered around the center of mass of the original hand depth image and hand shape features X.*

CNN_{hand_pose} outputs a vector J^* which contains 3D hand joints locations centered around the center of mass of the hand P_{com} in the original depth image. We can easily retrieve the original joint locations into the image space by applying an inverse transformation to the joints using P_{com}, as follows:

$$J_i = \{j_i^* + P^{com}\}_{i=1...K} \tag{5.2}$$

where j_i^* is the i^{th} joint in the predicted hand skeletal data J^*, P^{com} is the center of mass of the hand in the original depth image and J_i is the final skeletal data predicted by our CNN for hand pose estimation CNN_{hand_pose}.

From this network, we can extract two different feature vectors from a hand depth image: skeletal features J and a hand shape feature vector X lying in a high dimensional space.

5.2.5 CNN training procedure

Parameters of the model have to be initialized before the training step. A common way to generate the values is to use a random normal distribution. An exception is made for the bottleneck layer as we can help the network using prior knowledge. We initialize its weights with the 30 major components from a Principal Component Analysis (PCA) of the hand skeletal label space of the training set.

As the cost function, we minimize the Huber loss to evaluate the differences between the hand pose ground-truth J and the output of the network noted $\hat{J}$:

$$H(J, \hat{J}, \delta) = \begin{cases} \frac{1}{2}(J - \hat{J})^2 & \text{for } |J - \hat{J}| \leq \delta, \\ \delta\,|J - \hat{J}| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{5.3}$$

The Hubert loss is thus quadratic when the error is small ($\leq \delta$) and linear when it became larger. Consequently, this loss function is less sensitive to noisy annotations (which imply large errors) than the squared error loss function as depicted in Figure 5.5.

Weight decay is also applied to prevent an overfitting. The network is trained with back-propagation using the *Stochastic Gradient Descent* algorithm. The learning rate decrease following the number of epoch e by $\frac{lr}{1+0.2\times e}$.

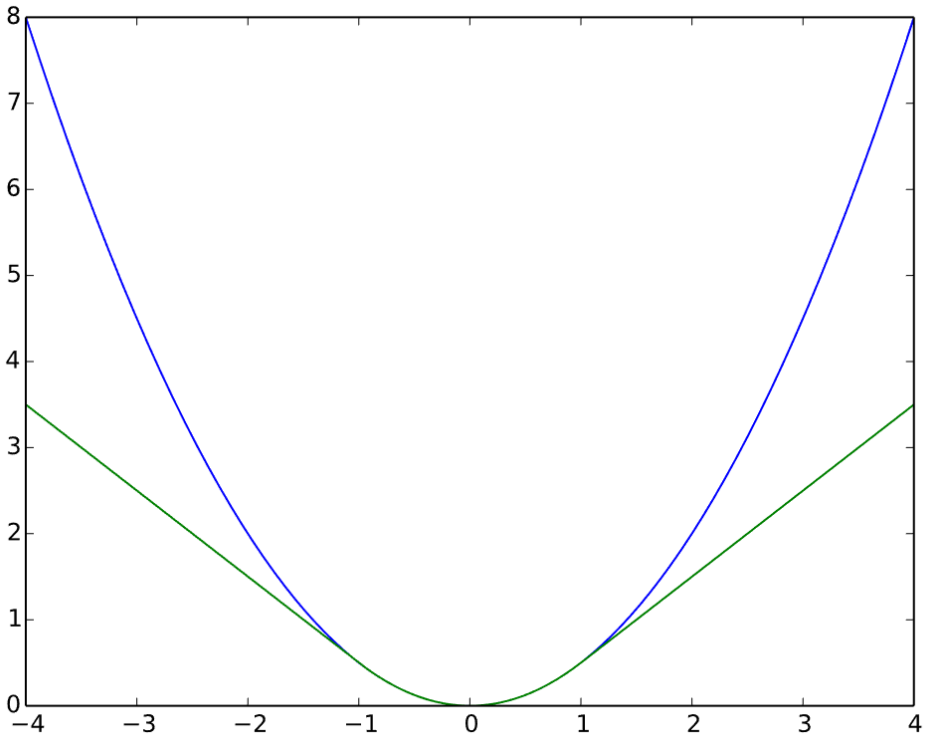


Figure 5.5 – *Huber loss defined in Equation 5.3 (green $\delta = 1$) and the squared error loss (blue $\frac{1}{2}x^2$) as functions of $J - \hat{J}$. Huber loss is less sensitive to large errors.*

5.3 Temporal features learning on hand posture sequences

A temporal modeling of hand skeletal feature sequences can efficiently be used for hand gesture recognition. Indeed, an algorithm which takes hand skeletal data as input is able to handle both *spatial* and *temporal* information. So, it has to extract efficient features to learn the dynamic aspect of both hand motions and shape variations. For example, it must learn that when a hand goes to the right, the user is performing a *Swipe Right* gesture but also that, if the user closed three fingers, it is, more precisely, a *Swipe Right with two fingers* gesture (e.g. contained in the NVIDIA dataset [94]).

To learn the temporal aspect of hand skeletal feature sequences, extracted using the CNN_{hand_pose}, we use LSTM layers. This recurrent neural network model has to learn the dynamic variation of both the hand shape and its motions in order to be efficient on the task of hand gesture recognition.

First, we use the hand pose estimation CNN_{hand_pose} to extract 3D joints

information from a hand depth image which describes the hand posture. Thus, hand gestures are represented as sequences of hand skeletal features $s = \{[j_1, \ldots, j_K]\}_{t=1\ldots N}$ be an ordered list of N vectors and $j_i \in J$.

To model the temporal aspect of gestures, we fed the hand skeletal sequences to two stacked LSTM layers with the purpose to map them into a single *spatial* and *temporal* description vector $h_t^{'}$, describing the temporal variations of the hand posture during the sequences.

Finally, recurrent layers are followed by a softmax layer which transforms the vector $h_t^{'}$ into a class-conditional probability $\hat{y}_t^i$ where $i \in \{1 \ldots K\}$ and K the number of gesture classes. The classification of a gesture is performed using the last output vector $\hat{y}_N$ of the sequence. Thus, the predicted class of the gesture is $\hat{y} = \underset{i}{\text{argmax}}(\hat{y}_N^i)$. This model, noted $RNN_{skeleton}$, is depicted in Figure 5.6.

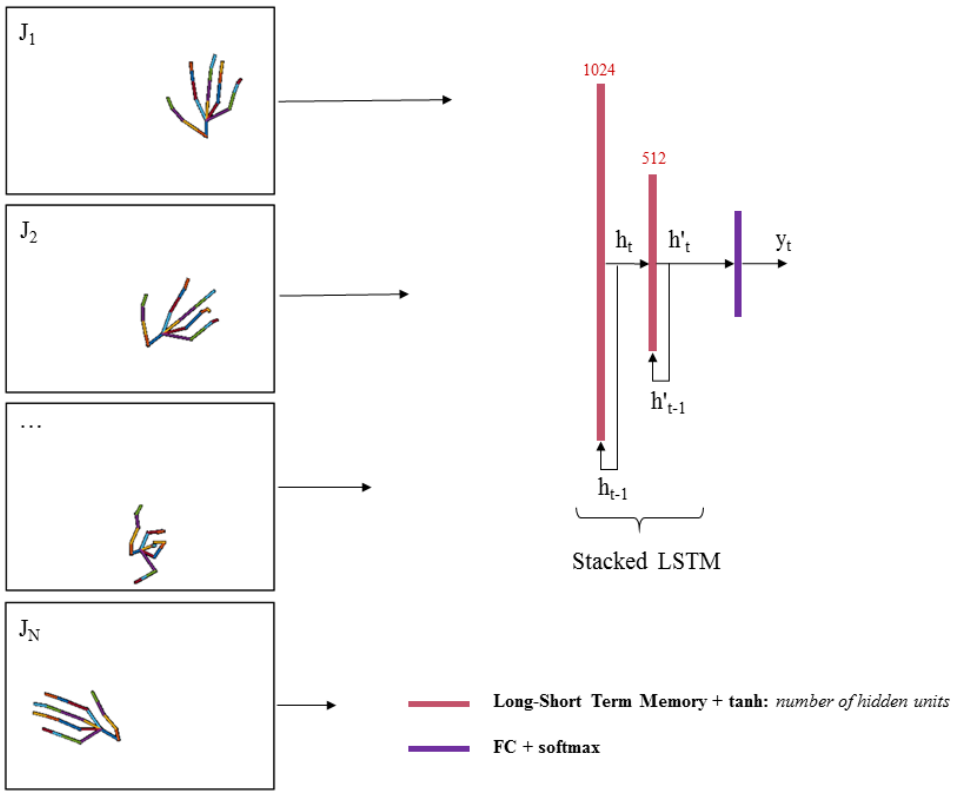


Figure 5.6 – *The network architecture which performs classification of hand gestures from temporal hand posture description using hand skeletal features. A sequence* $s = \{J_t\}_{t=1\ldots N}$ *is given as input to two stacked Long-Short Term Memory layers. Their output* $h_t^{'}$ *is finally transform by a fully connected layer which as a softmax activation function in order to output a class-conditional probabilities vector. The size of the last layer is not given in the figure as it is dependent to the number of class in the dataset. The final predicted label of the sequence s is* $\hat{y} = argmax_i(\hat{y}_N^i)$.

5.4 Temporal features learning on hand shape sequences

The method described in Chapter 3 confirms the importance of the hand shape information for hand gesture recognition algorithms. Indeed, precise and fine dynamic hand gestures are specially defined by hand shape variations though the gesture and less by its motions, such as fine gestures performed with fingers.

In this section, we describe a recurrent network model used to classify hand gesture sequences using exclusively the information of hand shape variations. We aim to improve the hand gesture recognition process by learning temporal descriptors specifically on hand shape feature sequences in addition to temporal hand skeletal features.

Before reducing the feature space to the skeleton space, the CNN_{hand_pose}, trained for hand pose estimation, has a transition state from which we can extract hand shape features, called X, laying in a high dimensional space: $X \in \mathbb{R}^{1024}$. In order to efficiently extract hand pose features, this state has to handle global and local hand shape information. Moreover, using this state as shape features allows to decrease the computing time of our framework as we do not need to compute additional descriptors.

First, we define a sequence of hand shape features $s = \{X_t\}_{t=1...N}$ which is an ordered list of N vectors and $X_t \in \mathbb{R}^{1024}$. Each vector X_t is extracted from a cropped hand depth image using the CNN_{hand_pose}.

Similarly to the $RNN_{skeleton}$, we create a new model, noted RNN_{shape}, composed of two stacked LSTM layers followed by a softmax prediction layer. We depict the operations performed by the RNN_{shape} in Figure 5.7.

5.5 Training procedure

During the training phase of both the $RNN_{skeleton}$ and the RNN_{shape}, weight decay and a dropout strategy are applied to prevent overfitting. Networks are trained using the Back-Propagation-Through-Time (BPTT) algorithm [160]. BPTT is equivalent of unrolling the recurrent layers, trans-

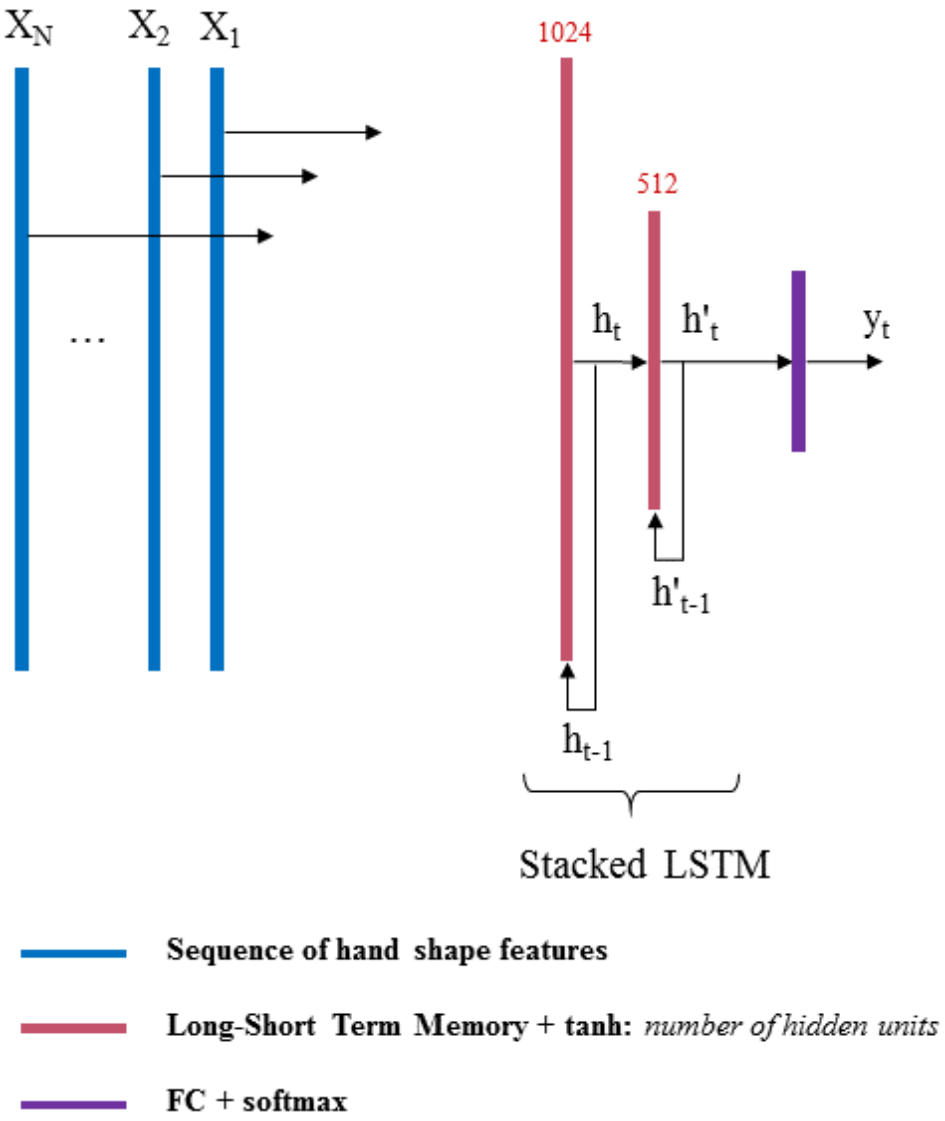


Figure 5.7 – *The network architecture to classify dynamic hand gesture using temporal hand shape variations features. A sequence* $s = \{X_t\}_{t=1...N}$ *is given as input to two stacked Long-Short Term Memory layers. Their output* $h_t^{'}$ *is finally transform by a fully connected layer which as a softmax activation function in order to output a class-conditional probabilities vector. The size of the last layer is not given in the figure as it is dependent to the number of classes in the dataset. The final predicted label of the sequence* s *is* $\hat{y} = argmax_i(\hat{y}_N^i)$.

forming them into a multi-layer feed-forward network of depth N; where N is the number of frame in the gesture. The standard gradient-based back-propagation, introduced in Algorithm 2 in Section 4.3.2, is then used. We average the gradients to consolidate weight updates to duplicated unrolling. The learning rate decreases following the number of epochs *ne* by $lr = 0.001 \times N_0 e^{-\lambda \times ne}$. Networks try to minimize the cross-entropy cost.

To increase variability in the training examples, we apply random horizontal, vertical and depth translations on depth image sequences before each learning iteration. This step is called *data augmentation*. Since recurrent connections can learn the specific order of gesture sequences in the training set, we randomly permute the training gesture videos before each new epoch.

5.6 Two-stream RNN fusion

The $RNN_{skeleton}$ encodes the hand posture variations and the RNN_{shape} encodes the shape variations, both, during the sequence. Once it is done, network outputs need to be fused.

In 2015, Jung *et al.* [57] proposed a joint fine-tuning strategy in order to join information from networks learning from different modalities and showed performance improvements. We aim to fuse the output of the $RNN_{skeleton}$ and the RNN_{shape} to enhance the classification process using a similar method as depicted in Figure 5.11.

Since both networks are trained separately, we retrain last fully connected layers before the softmax activation functions with a new cost function, noted $\mathcal{L}_{fusion}$, defined as follows:

$$\mathcal{L}_{fusion} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_2 + \lambda_3 \mathcal{L}_3 \tag{5.4}$$

where $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ are respectively loss functions computed on the $RNN_{skeleton}$, the RNN_{shape} and the sum of both outputs. The λ_1, λ_2 and λ_3 are tuning parameters. Each cost function is a cross entropy function. Let l_1 and l_2 be respectively the output values of the network $RNN_{skeleton}$ and RNN_{shape}, $\mathcal{L}_3$ is then defined as follow:

$$y_3^i = softmax(l_1^i + l_2^i) \tag{5.5}$$

$$\mathcal{L}_3 = -\frac{1}{N}\sum_{i=1}^{N}\left[y^i \log \hat{y}_3^i + (1-y^i)\log(1-\hat{y}_3^i)\right] \tag{5.6}$$

The final decision is obtained using y_3^i:

$$\hat{y} = \underset{i}{\text{argmax}}(y_3^i) \tag{5.7}$$

As a result, we utilize three loss functions in the training step: $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$. $\mathcal{L}_1$ and $\mathcal{L}_2$ are used to regulate, respectively, both streams and avoid that one of them vanished under the weight of the other. $\mathcal{L}_3$ is used to optimize the fusion of the two modalities. Consequently, we use only y_3 for prediction as it is impacted by both the RNN_{shape} and the $RNN_{skeleton}$. To train the joint fine-tuning method, we use the same training procedure defined previously.

5.7 Experimental results on the NVIDIA Hand Gesture dataset

In this section, we propose to analyze the impact of the RNNs separatly and fused on the recognition rate according to the type of gestures. We evaluate the performance of networks for the task of detection and recognition of hand gestures on the NVIDIA hand gesture dataset [94]. We finally compare our results and network capacities with the R3DCNN network [94].

5.7.1 Dataset

Very recently, Molchanov *et al.* [94] introduced a new challenging multimodal dynamic hand gesture dataset. Details of the dataset can be found in Section 3.8.4. We note that the dataset contains 1532 gestures of 25 hand gesture types depicted in Figure 5.8. The gestures include *Swipe* gestures, which are defined by hand motions, static gestures, as *Show up two fingers*,

which are only defined by hand shapes, or yet, *Rotation* gestures which need both information.

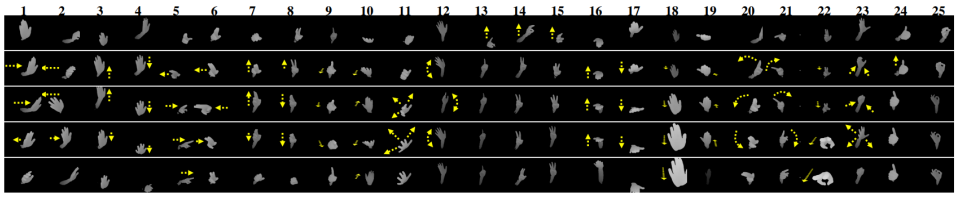


Figure 5.8 – *The 25 different gestures of the NVIDIA hand gesture dataset. Each column shows a different gesture class (1-25). The gestures included (from left to right: moving the hand left, right, up, or down; moving two fingers left, right, up, or down; clicking with the index finger; calling someone (beckoning with the hand); opening and shaking the hand; showing the index finger, two fingers or three fingers; pushing the hand up, down, out or in; rotating two fingers clockwise or counter-clockwise; pushing forward with two fingers; closing the hand twice; and showing "thumb up" or "Ok". The top and bottom rows show the starting and ending depth frames, respectively, and yellow arrows indicate the motion performed in intermediate frames. Image reproduced from [46].*

In the following experiments, the same protocol proposed by Molchanov *et al.* [94] is used: we randomly split the data into a training (70%) and a test (30%) sets, resulting in 1050 training and 482 test videos.

5.7.2 Implementation details

A) Hand posture and shapes features

To extract the deep features of hand posture and its shape, we train a CNN on the ICVL dataset [144], which comprises a training set of over 180,000 depth images showing various hand poses. The dataset is recorded using a time-of-flight *Intel Creative Interactive Gesture Camera* and has 16 annotated 3D joints for each depth image. Depth images have a high quality with little noise.

We use the Hubert loss function defined in Equation 5.3. We choose a sensitive factor to error $\delta = 500$ (we remind that 3D hand coordinates annotations are given in *mm*). It means that errors on the joint location prediction superior to half a centimeter is linear while smaller errors are quadratic.

Weight decay is applied with a regularization factor equal to 0.001. The networks are trained with a batch size of 128 for 100 epochs. The initial learning rate lr is set to 0.01 with a momentum of 0.9.

B) Temporal features

To avoid overfitting while training the recurrent layers, weight decay is applied with a regularization factor equal to 0.001. The dropout strategy has a probalistic value equal to 25%. We stop the training after 30 epochs to avoid learning training dataset specification. The initial learning rate lr is set to 0.001.

For the data augmentation step, the ranges of the horizontal and vertical translations are $\pm$20 pixels and the range of the depth translation is $\pm$100. Parameters for each translation are drawn from a uniform distribution.

5.7.3 Offline recognition analysis

A) Dynamic hand gesture recognition using temporal hand posture features

We propose to evaluate the $RNN_{skeleton}$ and to analyze its usefulness according to the type of gestures. We remind that the $RNN_{skeleton}$ aims to classify gestures using hand posture variations during the gestures hand skeletal features extracted by a CNN trained for hand pose estimation. We evaluate the coherence between predicted labels and ground-truths by computing the confusion matrix presented in Figure 5.9.

Using only the $RNN_{skeleton}$, we obtain an overall recognition rate of 65.41%, where the first statement is that only 5 gestures show an accuracy higher than 80%. We note that using hand skeletal features, inverse *Swipe gestures* (1 - 6) do not show too much confusion between them. Also *Rotation* gestures obtained a correct accuracy up to 80%.

Confusion between gestures *Swipe right hand open* (1) and *Swipe right one finger* (5) is up to 11%. In this case, the $RNN_{skeleton}$ recognizes the motion but struggles to define the hand shape differences. Also, the network does not perform correctly on almost every *Static gestures* (13 - 15, 25).

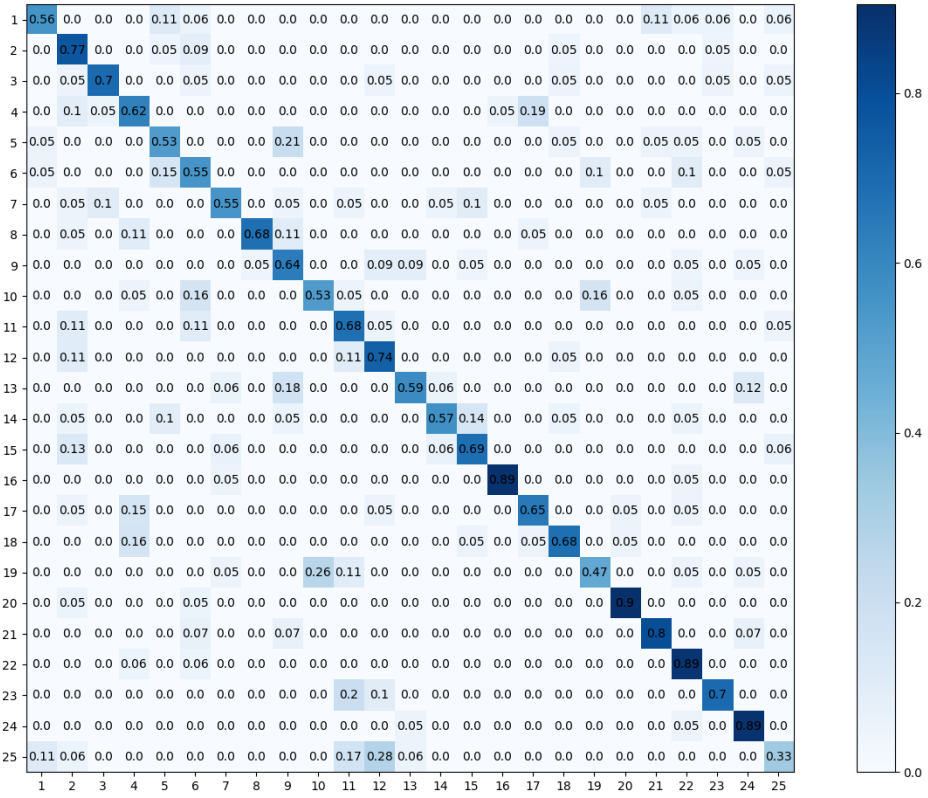


Figure 5.9 – *The confusion matrix obtained using the $RNN_{skeleton}$ on the NVIDIA Hand Gesture dataset. Labels of gestures can be found in Figure 5.8.*

B) Dynamic hand gesture recognition using temporal hand shape features

In this section, we propose to evaluate the RNN_{shape} and to analyze its recognition rates and its efficiency according to the type of gestures. We remind that the RNN_{shape} takes as input features based on the cropped region of interest of the hand and, so, does not contain any information about the absolute hand position in the scene.

Using only the RNN_{shape}, we obtain an overall recognition rate of 77.08%. It is 11.64% higher than the result obtained using the $RNN_{skeleton}$. Also, 11 gestures show an accuracy higher than 85%. From the resulting confusion matrix, we made several statements.

First, accuracies for gestures that are described by the hand shape (e.g. 12 - 14, 22 - 25) show the efficiency of using hand shape features coming from the high dimensional shape space of the CNN_{hand_pose}. Second, the need of the motion cues in hand gesture appears in the confusion matrix

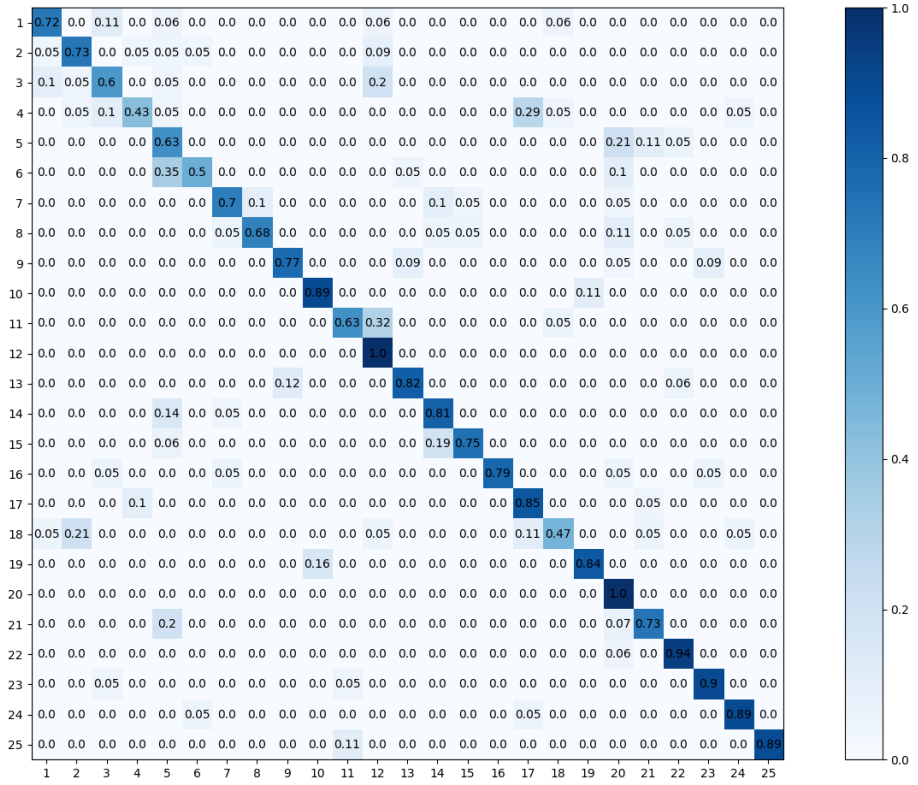


Figure 5.10 – *The confusion matrix obtained using the RNN_{shape} on the NVIDIA Hand Gesture dataset. Labels of gestures can be found in Figure 5.8.*

with lower results for several *Swipe* gestures (1 , 3 - 6, 18). In addition, gestures having similar hand shapes but different motion patterns as *Swipe left with the hand open* (2) and *Push the hand open* (18) show high confusions. Finally, some gestures with high similarities of hand shape, as *Show two fingers* and *Show three fingers*, still have high confusions and are a window for future improvements. Meanwhile, the RNN_{shape} does not show several misclassifications like those appeared using the $RNN_{skeleton}$ such as between gestures *Swipe right hand open* (1) and *Swipe right one finger* (5).

C) Fusion process

In Chapter 3, we studied a way to perform dynamic hand gesture recognition using the hand shape and its motions using handcrafted features on skeletal data. Results show the effectiveness of combining those two specific information to describe hand gestures.

Combining different information or modalities using deep learning is not an obvisous task. Wu *et al.* [162] investigated intermediate and late

fusion strategies for multimodal gesture recognition. They discover that averaging results in the last stage of the process gives accurate and more robust results. In this section, we evaluate multiple fusion configurations of the $RNN_{skeleton}$ and the RNN_{shape} depicted in Figure 5.11. If needed, new layers are trained using back-propagation. We keep using the *rmsprop* optimizer with a batch size of 10 for 30 epochs. The initial learning rate lr is set to 0.0001 and it decreases following the number of epochs ne by $lr = 0.001 \times N_0 e^{-\lambda \times ne}$. Model optimizers minimize the cross-entropy cost function during the training phase.

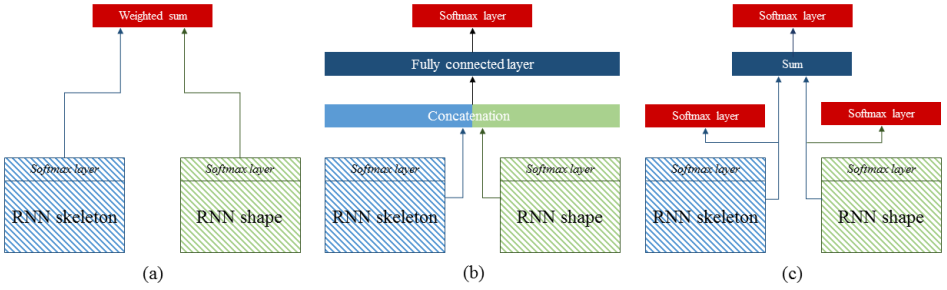


Figure 5.11 – *Three different fusion configurations: (a) Using a weighted summation of the final probabilistic distributions. (b) Adding fully connected layer to learn a representation of the two networks output. (c) Using the Joint Fine-Tuning methods introduced in Section 5.6. Hatched boxes are element that are fixed and not changed during the training fusion process.*

1 - Weighted summation of final probabilistic distributions. A strategy to fuse two modalities is to combine their final emission probabilities. Network outputs were integrated by weighted summing their values. Let be $\hat{y}_1$ and $\hat{y}_2$ be, respectively, the output probabilities of the $RNN_{skeleton}$ and the RNN_{shape}. In this fusion strategy, the predicted label are:

$$\hat{y}_{final} = \underset{i}{\text{argmax}}(\alpha \, \hat{y}_1^i \; + \; (1-\alpha) \, \hat{y}_2^i \;) \; , \; 0 < \alpha < 1 \tag{5.8}$$

with $i = 1 \ldots K$ and K is the number of classes. The parameter α depends on the performance of each network. We set the value of α to 0.48 which is the optimal value as shown in Figure 5.12. Using a weighted summation of the outputs, we obtain a final accuracy of 78.33%.

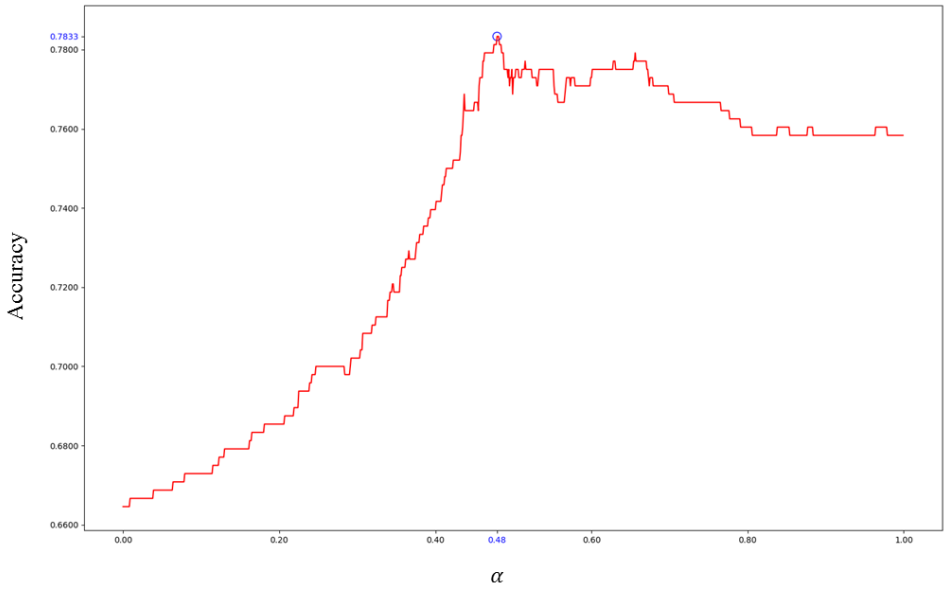


Figure 5.12 – *Performance of the fusion probabilistic distributions of the* $RNN_{skeleton}$ *and the* RNN_{shape} *using a weighted summation strategy with respect to* α*. We changed the value of* α *from* 0 *to* 1 *with interval of* 0.001*. We found an optimal value when* α *is equal to* 0.48 *with an overall accuracy of* 78.33%*.*

2 - Add and learn an additional fully connected layer. Another way to fuse two modalities is to learn a possible intermediate representation of output networks. We concatenate the outputs of the $RNN_{skeleton}$ and the RNN_{shape} and feed them into a fully connected layer followed by a softmax layer as depicted in Figure 5.11. It is similar to the intermediate solution investigate by Wu *et al.* [162]. Compared to them, this solution shows a better accuracy than simply apply a weighted sum with a final accuracy of 79.37%.

3 - Joint fine-tuning. The joint fine-tuning method introduced in Section 5.6 is a mixture of the two previous fusion strategies. It consists in retraining the two last softmax layers of the $RNN_{skeleton}$ and the RNN_{shape} while forcing their sum to be a representation of both networks. This strategy takes advantage of previous fusion method benefits. Firstly, it does not add parameters to the model and, so, does not increase its complexity. Secondly, it allows the network to learn a joint representation of network outputs. Using this approach, the final accuracy of fusion networks is 79.58%. The confusion matrix is given in Figure 5.13.

The confusion matrix shows that results of both RNN have not de-

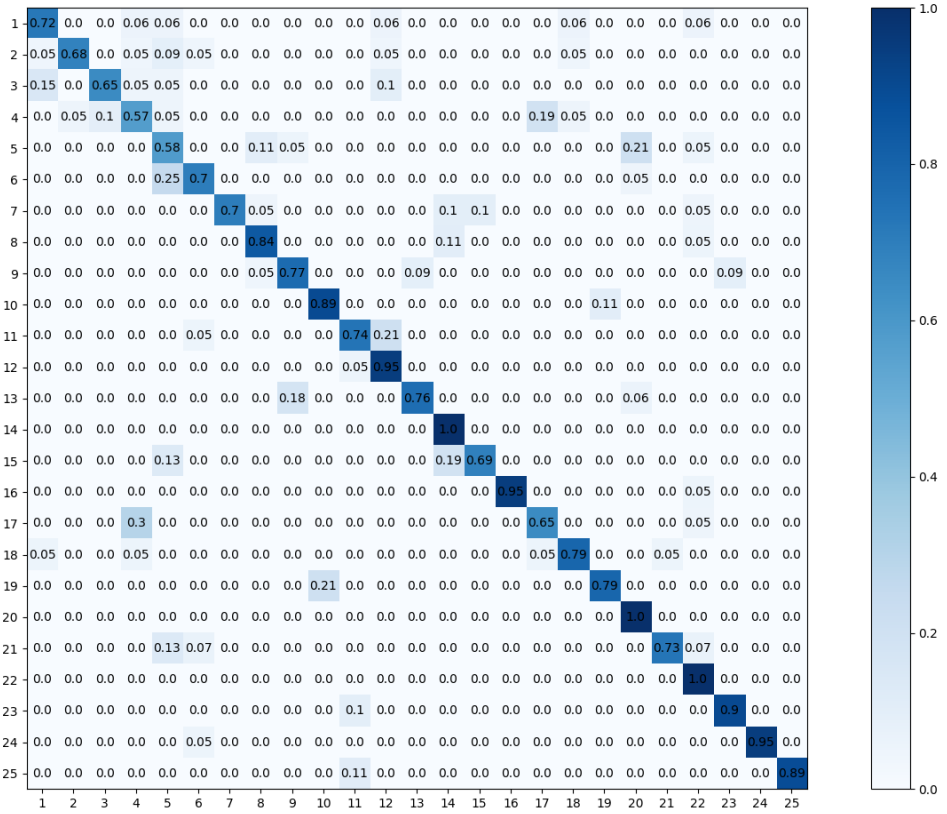


Figure 5.13 – *The confusion matrix obtained after the fusion by joint-fine tuning of the RNN_{shape} and $RNN_{skeleton}$ on the NVIDIA Hand Gesture dataset. Labels of gestures can be found in Figure 5.8.*

creased and that the recognition system benefits from the fusion. Let us take as example the gesture *Show up two finger* (index 14 in Figure 5.8). Using the RNN_{shape}, it shows high confusion (14%) with the gesture *Swipe right with two fingers* (5) as the network does not take motion information into account. Using the $RNN_{skeleton}$, the gesture *Show up two finger* shows, this time, high confusion (14%) with the gesture *Show up three finger* (15) as the network do no handle precise information about the hand shape. After the fusion step, the recognition accuracy of the gesture *Show up two finger* reach 100%.

5.7.4 Online detection of continuous hand gestures

The difference between offline and online recognition is in the mapping definition. Offline recognition maps a sequence to a class-conditional probability vector y following a mapping function $F_{offline}$, in our case, defined as:

$$F_{offline} : \{I_t\}_{t=1...N} \mapsto y \tag{5.9}$$

where $\{I_t\}_{t=1...N}$ is a sequence of depth images of size N. Instead, an online recognition algorithm maps each frame of the sequence to a class-conditional probability vector:

$$F_{online} : \{I_t\}_{t=1...N} \mapsto \{y_t\}_{t=1...N} \tag{5.10}$$

In this section, we study the online potential of the framework introduced so far. Online hand gesture recognition requires a prior gesture detection as it contains motions belonging to none of the gesture categories. We study two different methods to perform gesture detection and new metrics to analyze the online capacity are introduced.

A) Gesture detection

An unsegmented stream of gestures contains a lot of unwanted and meaningless hand motions that do not belong to none of the gesture categories. First, hand gesture movements are often composed of three phases:

1. The *pre-stroke* phase, which is composed of hand motions happening before the relevant gesture when the user needs to put its hand in a starting position. For example, it is the movement the user performs to move the hand from its restful position to a place where the camera can see the hand.

2. The nucleus phase, where the hand gesture is performed and have meanings.

3. The *post-stroke* phase, which is composed of hand motions happening after the relevant gesture when the user wants to move back its hand to a restful position.

Additionally, a stream of gestures contains motions between the gestures as depicted in Figure 5.14. For example, in a human-computer interface based on hand gestures in a car scenario, while the user is not performing a gesture, his hands are still moving to control the vehicle

and, so, contains a lot of parasitical hands motions. A challenge of online hand gesture recognition is to detect and extract only hand motions from nucleus phases in order to improve the gesture recognition accuracy.

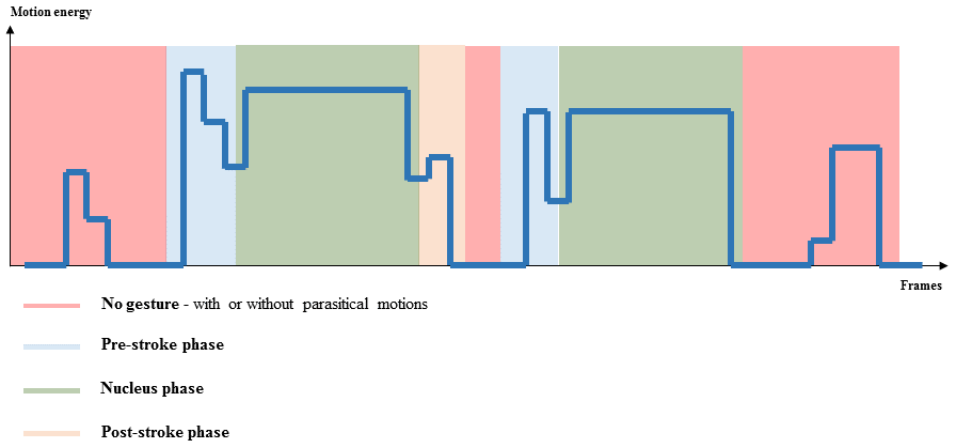


Figure 5.14 – *Different phases of a continuous stream of gestures. The figure depicts a fictive curve of the motion energy of a sequence of two hand gestures. Frames which do not belong to a gesture, with or without parasitical motions, are in red. Blue, green and orange squares identify, respectively, pre-stroke, nucleus and post-stroke phases. We note that the second gesture does not contain a post-stroke phase as with the pre-stroke phase, their existence is not automatic.*

In the following experimentation, we study the gesture detection step of our framework and its impact on the overall accuracy. The NVIDIA dataset has been captured following a human-computer interaction based on hand gestures in a car scenario. While the user is not performing a gesture, its hands still move to control the vehicle and, so, is highly suitable to study gesture detection.

The ground-truth annotations of the NVIDIA dataset do not contain the nucleus locations, we manually labeled each frame following a binary categorical variable $\{gesture, no\_gesture\}$. To compare our detection step, we first remove every frame labeled *no_gesture* of the dataset and perform the whole classification process as defined so far. We obtain an overall accuracy of 83.33%, which outperformed the previous score by 3.75%. This result shows the importance to perform gesture detection. We compare two ways to accomplish the detection step: by adding a gesture localization step using a frame-wise light gesture localization network and by adding a garbage class that gathers all *no_gesture* motions.

1 - Adding a frame-wise light gesture localization network. In real applications, we do not have information about when and where the hand gesture is going to be performed. Neverova *et al.* [98] added a binary classification step before the classification process using {*gesture*, *no_gesture*} labels. Following this method, we propose to learn a light binary classifier network composed of a single LSTM layer, but the network is trained based only on skeletal features. The localization network is placed before the classification network. If the localization network classifies the current frame as *gesture*, it is fed to the classification network, it is rejected otherwise.

On the task of frame-wise binary classification following labels {*gesture*, *no_gesture*}, our localization network has obtained 76.31%. It is a weak result for a task with only two classes. To compare, Neverova *et al.* obtained a binary classification up to 98% on the *Chalearn 2014 Looking at People Challenge (track 3)* [33] using body skeletons. The aim of this track is to perform classification on a dataset of 20 Italian conversational gestures. These are not hand gestures but whole body gestures and they are mostly defined by arm movements. Between gestures, users are not moving and keep the same body restful position making the gesture localization task much easier.

In hand gestures, there are high similarities between some *pre-stroke* phases and other gestures nucleus phases. Figure 5.15 shows the frame-wise classification following labels {*gesture*, *no_gesture*} by gesture types of the NVIDIA dataset using our localization network.

The first thing we observe is that static gestures (e.g. *Show up one, two or three fingers* (12 - 16) or yet *Ok sign* (24) and *Thumb up* (25)) show accuracy higher than 80%. An other interesting statement is that *Swipe left* gestures (1 and 5) show weak results, respectively, 61.59% and 72.43%. While, *Swipe right* gestures (2 and 6) reach accuracies higher than 80%. Indeed, as depicted in Figure 5.16, different phases of inverse gestures contain high similarities. For example, the pre-stroke phase of a *Swipe left* gesture consists in moving the hand to the right so that the camera is able to see the entire gesture. However, this movement to the right can be seen

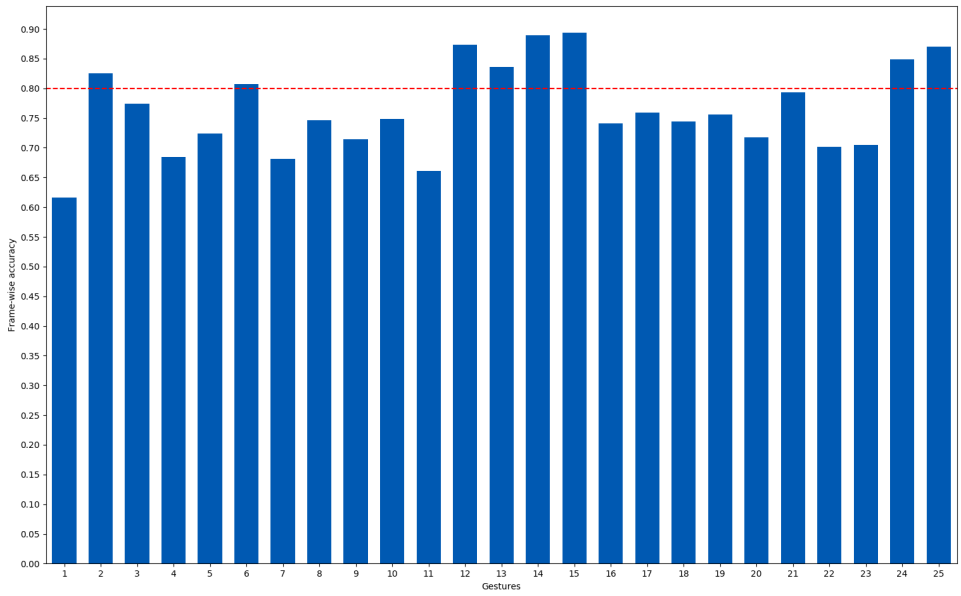


Figure 5.15 – *Accuracies of correctly classified frames following labels {gesture, no_gesture} by gestures of the NVIDIA dataset and using a light network composed of one LSTM layer. The red line shows the limit at 80% of recognition.*

as a *Swipe right* gesture nucleus by the localization algorithm and not as a pre-stroke phase of a *Swipe left* gesture.

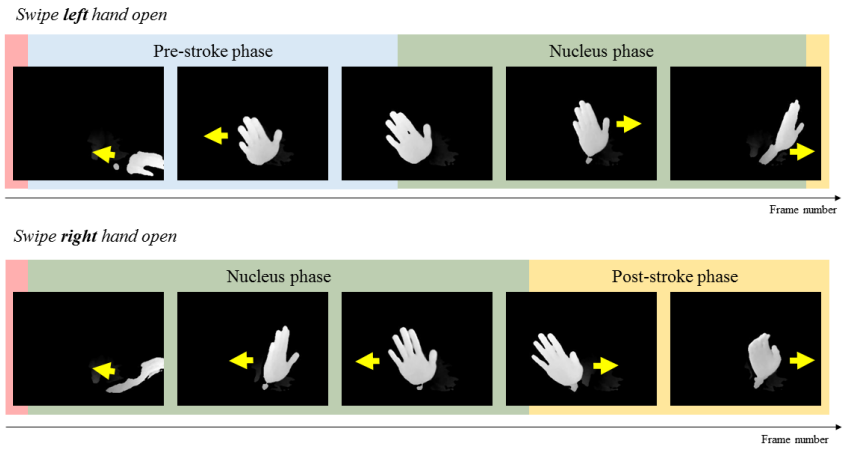


Figure 5.16 – *An example of a gesture Swipe Left (up) and Swipe Right (down), both hand open, and their respective phases (blue: pre-stroke, green: nucleus, orange: post-stroke). The figure shows high similarities between the pre-stroke phase of the Swipe Left gesture and the nucleus phase of Swipe Right gesture.*

Despite a bad recognition rate of nucleus phases by a localization network, we went through the entire recognition process. Compared to our previous results, we obtain a lower overall recognition accuracy equal to 78.87% on the NVIDIA dataset.

2 - Adding a garbage class. This method consists of extending the dictionary of existing gestures such as: $Y' = Y \cup \{ \textit{no\_gesture} \}$. Consequently, the softmax layer outputs a class-conditional probability for this additional "garbage" class. All frames which do not belong to a nucleus phase are labeled with this new class.

To compare the effectiveness of the "garbage" class against the method which add a binary classification step, frames predicted as belonging to any gestures are mapped as positive examples, others are considered as negatives. Each frame is thus assigned with a label *gesture* or *no_gesture*. On the task of frame-wise binary classification following labels $\{gesture,\ no\_gesture\}$, the "garbage" class strategy obtained 84.72%, being 5.85% more accurate than using an additional binary classification step.

Figure 5.17 shows results of binary frame-wise accuracyies using a "garbage" class by gesture types of the NVIDIA dataset. We observe that 12 gestures show a score higher than 80%. As it is more accurate, we apply the "garbage" class strategy to our framework in order to perform online hand gesture recognition.

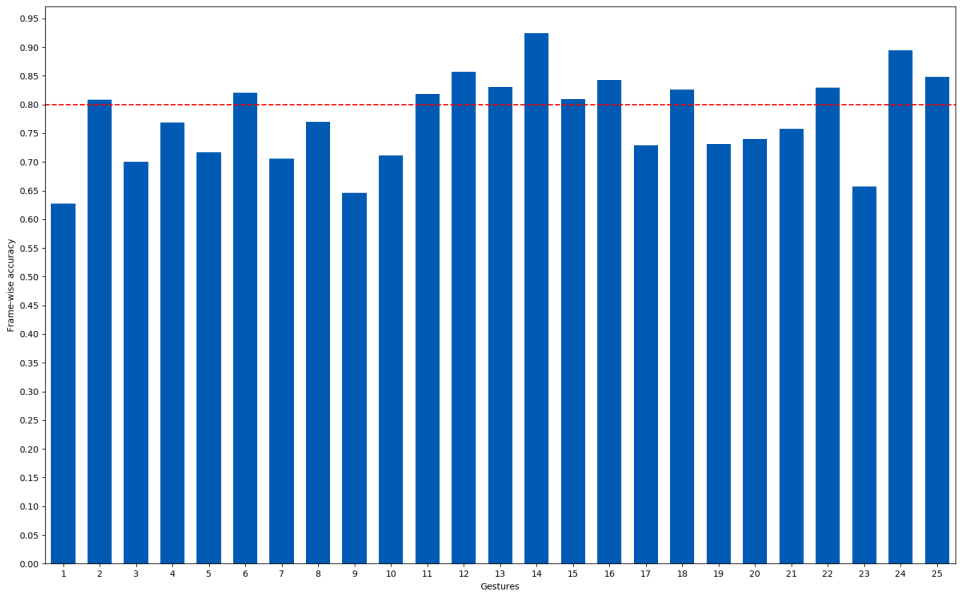


Figure 5.17 – *Accuracies of correctly classified frames following labels* $\{gesture,\ no\_gesture\}$ *by gesture types of the NVIDIA dataset using a "garbage" class. The red line shows the limit at* 80% *of recognition.*

B) Metrics for online gesture detection and recognition algorithms

In the folowing section, we analyze the online gesture detection and recognition capacity of our framework applied to the NVIDIA dataset. To do so, we use three metrics: the Receiver Operating Characteristic (ROC) curve [11] and the Normalized Time to Detect (NTtD) [51] for the detection analysis and the recognition accuracy to analyze the recognition process.

1 - ROC curve. We are considering testing a detector on a set of time series. The False Positive Rate (FPR) of the detector is defined as the fraction of time series that the detector fires outside of the event of interest (in our case the nucleus). The True Positive Rate (TPR) is defined as the fraction of time series that the detector fires inside a nucleus phase. A detector typically has a detection threshold that can be adjusted to trade off high TPR for low FPR and vice versa. By varying this detection threshold, we can generate and plot a ROC curve, which is the function of TPR against FPR. We use the area under the ROC for evaluating the detector accuracy.

2 - NTtD. To evaluate the detection time, we used the NTtD. Given a testing hand gesture where the nucleus occurs from a to b. Suppose the detector starts to fire at time t. For a successful detection, $a \leq t \leq b$, NTtD is the fraction of the nucleus that has occurred before the system fires a detection:

$$\frac{t-a+1}{b-a+1} \tag{5.11}$$

By adjusting the detection threshold chosen using the ROC curve, one can achieve lower NTtoD at the cost of higher FPR and vice versa.

3 - Recognition accuracy. The recognition accuracy is the fraction of sequences in the test set which is correctly labeled by our framework. Previously, for offline gesture recognition, we choose the predicted label of the last frame as the predicted label of the sequence. In an online scenario, we can not perform gesture classification in that way as it is possible that

the last frame is labeled as *no_gesture* and that does not implies a wrong classification.

Consequently, to compute a recognition accuracy for an online hand gesture recognition, the predicted label of a particular gesture is chosen as the predicted label of the last frame which is not labeled as *no_gesture*, such as:

$$\hat{y} = \underset{i}{\text{argmax}}(\ y^i_M\ ) \mid \underset{i}{\text{argmax}}(\ y^i_{M+1\ldots N}\ ) = no\_gesture \tag{5.12}$$

where N is the number of frame in the gesture.

C) Evaluation results

In the following experiments, we evaluate the effectiveness of our approach of hand gesture detection and recognition on the NVIDIA dataset.

To detect the presence of any one of the 25 gestures relative to *no_gesture*, we compare the highest current class probability output of our framework to a threshold $\xi \in [0, 1]$. When the detection threshold is exceeded, a classification label is assigned to the most probable class. First, we do it in a frame-wise manner and compute the ROC curve depicted in Figure 5.18.

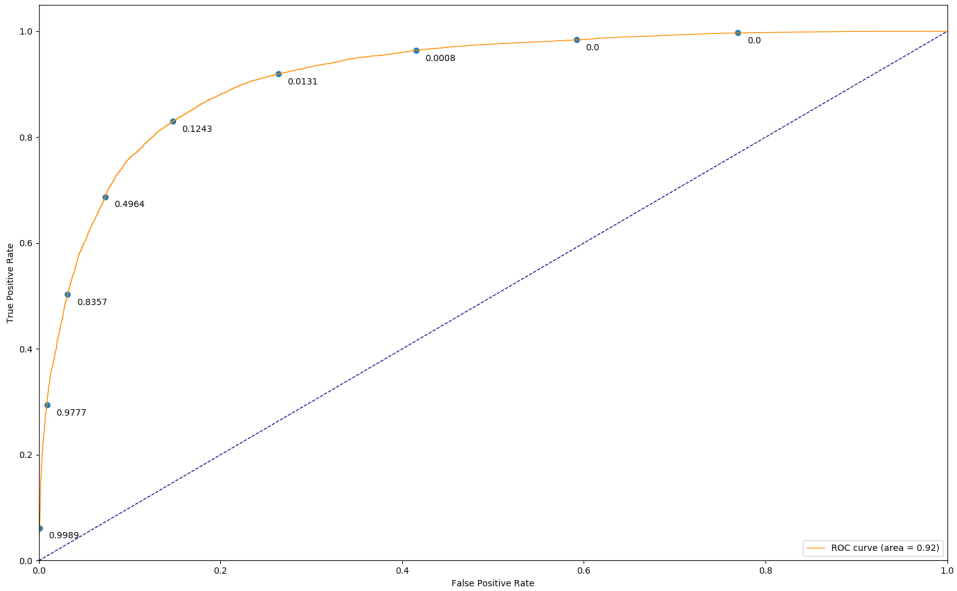


Figure 5.18 – *The ROC curve using our framework on the NVIDIA dataset. We obtained an Area Under the Curve equal to* 0.92.

Using it, we choose a detection threshold ξ equal to 0.16 as it shows a good trade-off between a high TPR (85%) and a low FPR (17%). The NTtD distribution values for various gesture types is shown in Figure 5.19. The average NTtD across all classes is 0.2158 which means that, in average, a hand gesture can be detected after only 22% of its nucleus. In general, static gestures require the finest portion of the nucleus to be seen before classification (around 10%), while dynamic gesture are classified on average within 25%.

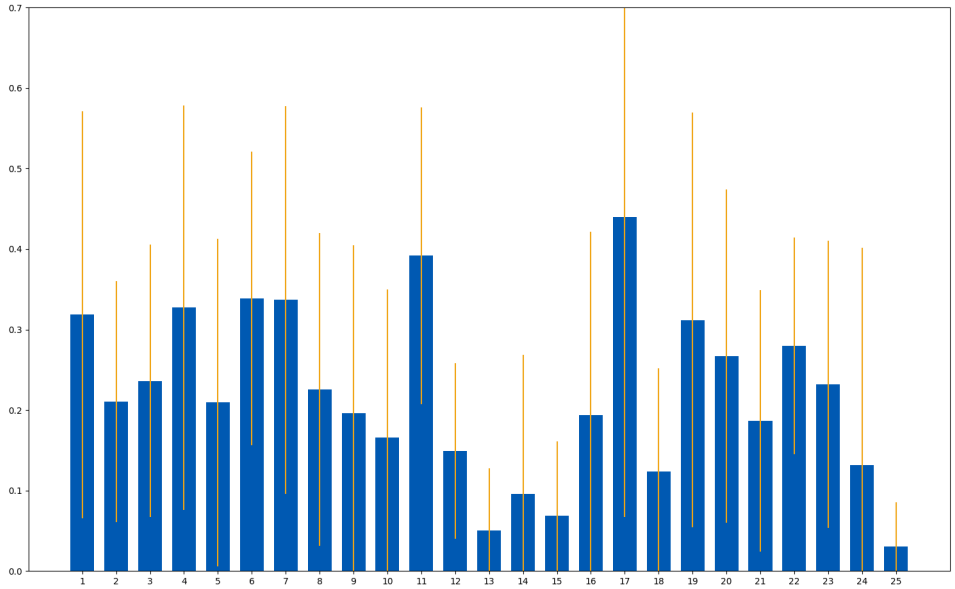


Figure 5.19 – *The Normalized Time to Detect values for the 25 gestures contained in the NVIDIA dataset using our framework and a detection threshold ξ equal to* 0.16.

The average nucleus length over the whole dataset are given in Figure 5.20.

Static gestures have longest nucleus phases. Intuitively, NTtD differences between dynamic and static gestures are explained as users letting their hand a long time in front of the camera to express a static gesture but the algorithm can detect it using few frames.

Finally, we compute the overall recognition accuracy using the fusion of the $RNN_{skeleton}$ and the RNN_{shape} in addition to a "garbage" class to perform an online hand gesture recognition. We obtained an accuracy of 81.25%. The confusion matrix is given in Figure 5.21.

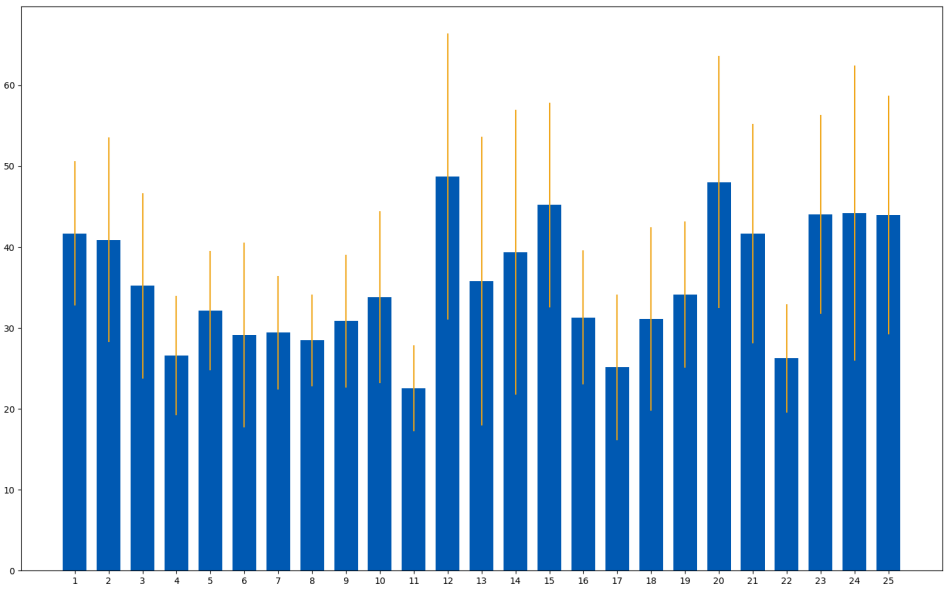


Figure 5.20 – *Nucleus lengths in number of frames of the 25 gestures contained in the NVIDIA dataset.*

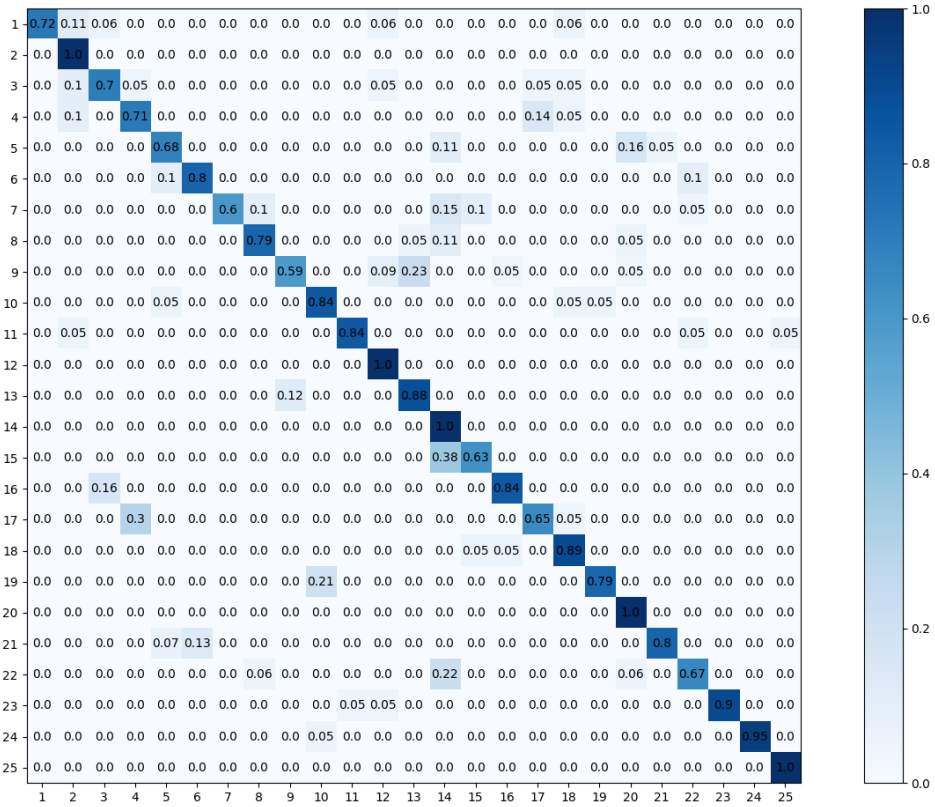


Figure 5.21 – *Final confusion matrix using the $RNN_{skeleton}$ and the RNN_{shape} fused in addition with a "garbage" class to performed online hand gesture recognition. The average recognition accuracy is equal to* 81.25%.

Table 5.1 – *Comparison with state-of-the-art methods on the NVIDIA dataset.*

Method	**Modality**	**Features extraction strategy**	**Accuracy**
Human	color		88.4%
HOG+HOG^2 [105]	depth	handcrafted	36.3%
SNV [167]	depth	handcrafted	70.7%
C3D [149]	depth	learned	78.8%
R3DCNN [94]	depth	learned	80.3%
Ours	depth	learned	**81.3%**

D) Comparison with state-of-the-art methods

We compare our approach to several state-of-the-art methods: HOG+HOG^2 descriptors [105], Super Normal Vector (SNV) [167], convolutional 3D (C3D) [149] and a C3D followed by a recurrent layer (R3DCNN) [94], as well as human labeling accuracy.

To compute HOG+HOG^2 descriptors [105], all video sequences are ressampled to 32 frames and the parameters of the SVM classifier are tuned via grid search to maximize accuracy. Among the CNN-based methods, we compare against the C3D method [149], which is pre-trained with the Sports-1M [59] dataset and fine-tuned with the depth modalities of the NVIDIA dataset. The R3DCNN method uses the C3D network to extract spatiotemporal features of sub-video clip of 8 frames and fed the result sequence in a recurrent layer. Molchanov *et al.* [94] trained the whole network using a Connectionist Temporal Classification (CTC) [42, 43] loss function.

Lastly, Molchanov *et al.* [94] evaluated human performance on the NVIDIA dataset by asking six subjects to label each of the 482 gestures videos in the test set after viewing the front-view color video. Prior to the experiment, each subject familiarized themselves with all 25 gesture types. State-of-the-art method results are given in Table 5.1. We note that handcrafted methods give lower results that deep learning methods. Our approach achieves the best performances, meanwhile it is still below human accuracy (88.4%).

Focus on the number of parameters in networks. In Chapter 4, we defined the capacity of a model following its size and its depth. Higher

Table 5.2 – *Formulas of the number of parameters for different layers with a number of hidden parameters equal to n and an input of size m.*

Layers	Formulas	Examples (m = 64, n = 9)
Fully Connected Layer	$m \times n$	576
Recurrent layer	$m \times n + n^2$	657
Long Short Term Memory	$4 \times (m \times n + n^2)$	2628
CNN	$m \times n$	576

are the size and the depth, higher is the number of parameters. Differences in the computational complexity between models are not exactly linearly comparable to their number of parameters, as some layers can see their computational time decrease dramatically using parallel computing (e.g. convolutional layer). However, it is a good start to study the overall complexity differences between models. Formulas giving the number of parameters of different layers are shown in Table 5.2.

The R3DCNN [94] contains 79,116,288 parameters distributed as follows: they extract spatiotemporal features using a 3D CNN of 8 convolutional steps and two fully connected layers of size 4096 which together contains 77,885,776 parameters. They append a recurrent layer of size 256 (1,114,112 parameters) and a softmax layer (6,400 parameters).

Our framework extracts hand shape and skeletal features from a single light 2D CNN with 3 convolutional steps and two fully connected of size 1024 which together contains 3,182,414 parameters. Our method uses also two-stacked LSTM layers, both, containing 14,680,064 parameters and end on a single softmax layer of 12,800 parameters. The whole framework described in this chapter contains 32,555,342 parameters, so, less than half the number contained in the R3DCNN [94] network and still outperforms their accuracy result by 1%.

The tranfer strategy using a hand pose estimation system to extract hand shape and skeleton features allowed us to perform better while using a far less complex network.

5.8 Experimental results on the Online Dynamic Hand Gesture (Online DHG) dataset

In Section 3.2, we introduced the Dynamic Hand Gesture - 14/28 (DHG-14/28) dataset that we have created to study the potential of using skeletal features for heterogeneous hand gesture recognition. This dataset is composed of 2800 sequences, each corresponding to a labeled gesture, providing, both, depth image sequences and hand skeletal features. Therefore, as the sequences are pre-segmented, the DHG dataset is not suitable to study the potential of our framework to perform gesture detection.

To study the potential of our framework on challenges coming from both the detection of gestures in an unsegmented video stream and the recognition process of heterogeneous hand gesture types, we created a new version of the DHG-14/28 dataset called *Online Dynamic Hand Gesture* (Online DHG) dataset. We note that for all following experiments, we use the same implementation details as defined in Section 5.7.

5.8.1 Dataset

All sequences of the Online DHG dataset are newly recorded and are different from the first version DHG-14/28 dataset. The dataset has been designed to keep all the characteristic of its first version: it contains the same 14 coarse and fine gesture types (presented in Table 5.3). Each gesture is either performed using one finger or the whole hand. Both depth image sequences and hand skeletal features are recorded by the Intel RealSense camera [117]. Further details are provided in Section 3.2.

Beside, the Online DHG dataset is different as it provides 280 sequences of 10 unsegmented gestures occurring sequentially. Between meaningful gestures, participants were free to take back a restful position without any suggestions from the recorders. In addition, comparing to the sequence-wise labeling of the first version, we provide a label for each frame of the gesture sequences. Frames between meaningful gestures are labeled as belonging to a *no gesture* class. Providing unsegmented sequences of many distinct gestures with frame-wise labeling will allows

Table 5.3 – *Gesture list included in the Online DHG dataset.*

Index (14)	Index (28)	Gesture	Labelization
1	1 2	Grab	Fine
2	3 4	Expand	Fine
3	5 6	Pinch	Fine
4	7 8	Rotation CW	Fine
5	9 10	Rotation CCW	Fine
6	11 12	Tap	Coarse
7	13 14	Swipe Right	Coarse
8	15 16	Swipe Left	Coarse
9	17 18	Swipe Up	Coarse
10	19 20	Swipe Down	Coarse
11	21 22	Swipe X	Coarse
12	23 24	Swipe V	Coarse
13	25 26	Swipe +	Coarse
14	27 28	Shake	Coarse

researchers to use the Online DHG dataset to study the potential of their algorithms to perform gesture detection and online recognition. We randomly split the dataset into a training (70%) and a test (30%) sets, resulting in 196 training and 84 test unsegmented videos.

5.8.2 Offline recognition analysis

First, we aim to experiment the potential of our framework to perform an offline recognition. We analyze the results obtained using, first, the $RNN_{skeleton}$ and the RNN_{shape} separately and finally the benefits coming from their fusion. To study the offline recognition process, we extract the gesture nucleus resulting in a dataset of 2800 gestures of either 14 or 28 distinct labels.

A) Dynamic hand gesture recognition using hand posture features

We propose to evaluate the $RNN_{skeleton}$ for the task of hand gesture recognition from the 14 gesture vocabulary of the Online DHG dataset. We remind that the $RNN_{skeleton}$ aims to classify gestures using hand posture variations along the time represented by hand skeletal sequences.

Using only the $RNN_{skeleton}$, we obtain an overall recognition rate of 84.52% where the first statement is that 10 gestures show an accuracy higher than 80%. Figure 5.22 illustrates the confusion matrix obtained for this experiment.

We note that using hand skeletal features, *coarse* gestures (6 - 14) do not show too much confusion between each other. In the meantime, recognition accuracies of *fine* gestures are the lowest. For example, the confusion between the *Grab* (1) and the *Expand* (2) gestures is up to 23%.

In our framework, we first extract the skeletal features from a pre-trained CNN using formulations in Section 5.2. The Intel RealSense [117] camera is also able to extract such features. We trained again the $RNN_{skeleton}$ from scratch , using skeletal features captured this time from the Intel Realsense camera. We obtain an overall accuracy equal to 84.88%, showing that the temporal modeling of our approach is independent of the skeletal features used.

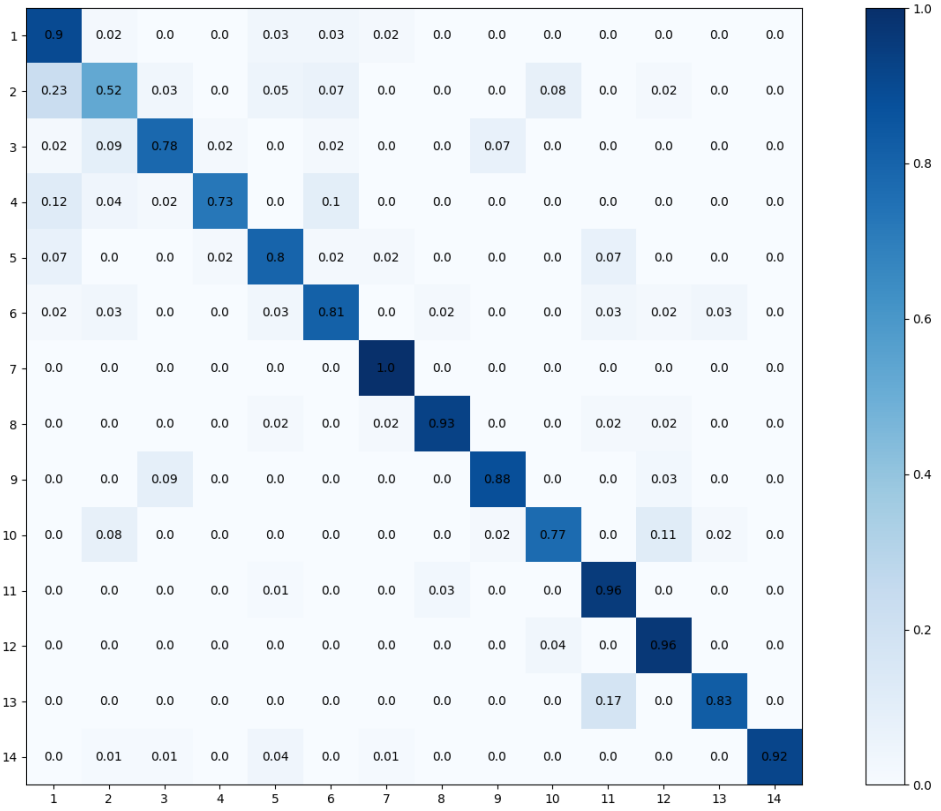


Figure 5.22 – *The confusion matrix obtained using the $RNN_{skeleton}$ on the Online DHG dataset for the task of offline recognition of 14 gesture types. Labels of gestures can be found in Table 5.3.*

B) Dynamic hand gesture recognition using hand shape features

We propose to evaluate the RNN_{shape} for the task of hand gesture recognition from the 14 gesture vocabulary of the Online DHG dataset. The RNN_{shape} takes as input features based on the cropped region of interest of the hand and, so, does not contain any information about the absolute hand position in the scene.

Using only the RNN_{shape}, we obtain an overall recognition rate of 80.60%. Eight gestures show an accuracy higher than 80%. From the resulting confusion matrix, depicted in Figure 5.23, we made several statements.

First, as the RNN_{shape} has no cues about the hand motions, several *Swipe* gestures show poor results as the gestures *Swipe Down* (10), *Swipe X* (11), *Swipe +* (13). Beside, the recognition accuracy of gestures that are described more by the hand shape variations (1 - 5) increases compared to the accuracy obtained using the $RNN_{skeleton}$. Those results confirm the

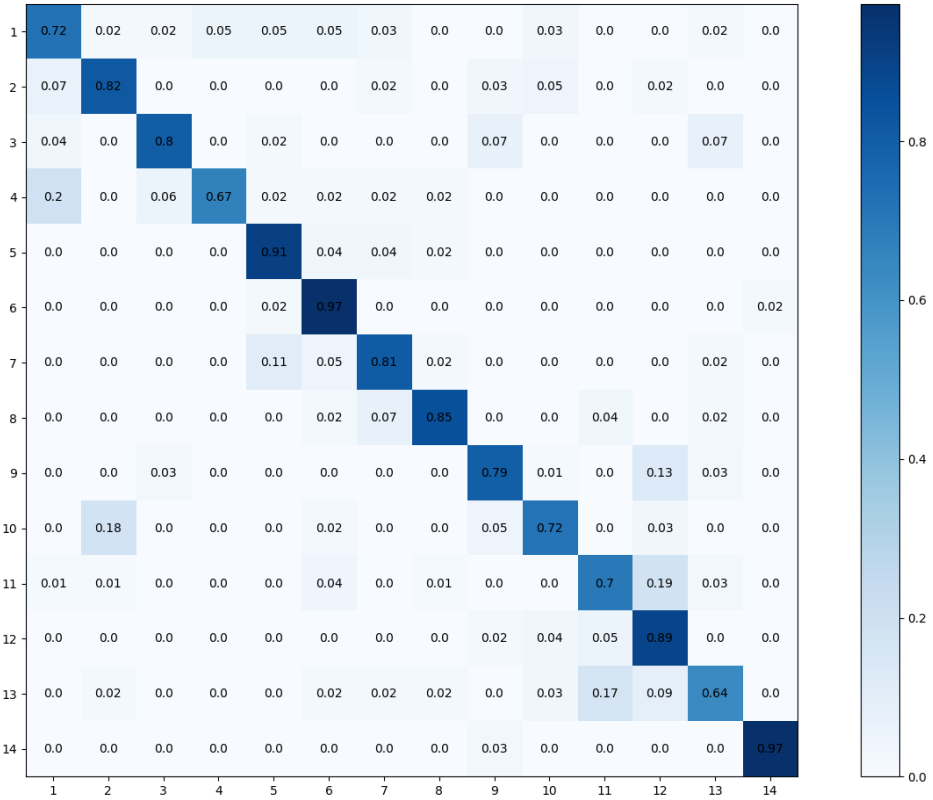


Figure 5.23 – *The confusion matrix obtained using the RNN_{shape} on the Online DHG dataset for the task of offline recognition of 14 gesture types. Labels of gestures can be found in Table 5.3.*

statement we made in Section 5.7.3 and show again the efficiency of using the high dimensional shape space of the CNN_{pose} ashand shape features to distinguish *fine* gestures.

C) Fusion process

Finally, we fuse the two *RNNs* and obtain an overall accuracy of 94.17%. The confusion matrix is depicted in Figure 5.24.

The matrix illustrates that the fusion is not only able to choose the right features to perform the classification but also takes benefit from both information to outperform the previous recognition accuracies. For example, the gesture *Swipe Down* (10) obtained 77% and 72% of accuracy using respectively the $RNN_{skeleton}$ and the RNN_{shape}. Once they are fused, our system is able to correctly recognize 89% of these gestures.

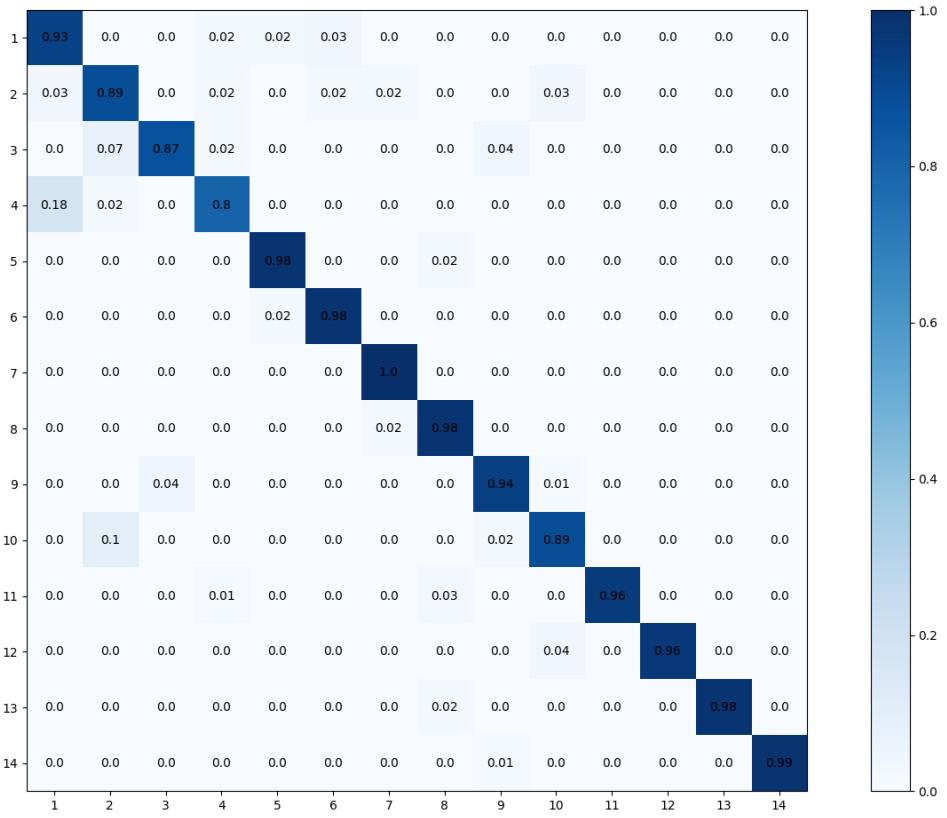


Figure 5.24 – *The confusion matrix obtained using a joint-fine tuning fusion of the RNN_{shape} and the $RNN_{skeleton}$ on the Online DHG dataset for the task of offline recognition of 14 gesture types. Labels of gestures can be found in Table 5.3.*

Let us now study the capacity of our framework to distinguish the same gestures performed with one finger or the whole hand. With a vocabulary of 28 gesture types, we obtain an accuracy of 76.30% and 76.67% respectively using the $RNN_{skeleton}$ and the RNN_{shape}. Once fused, we obtain an overall accuracy equal to 90.48%. This result illustrates the outstanding potential of fusing shape and posture features to perform fine hand gesture recognition. The confusion matrix resulting from this experiment, depicted in Figure 5.25, show that we obtain almost no confusion between gestures with the same meaning but performed with different number of fingers, thanks to the precise shape information coming from the RNN_{shape}.

D) Comparison with state-of-the-art methods

We compare here our framework to state-of-the-art methods for the task of offline hand gesture recognition on the Online DHG dataset. First, with

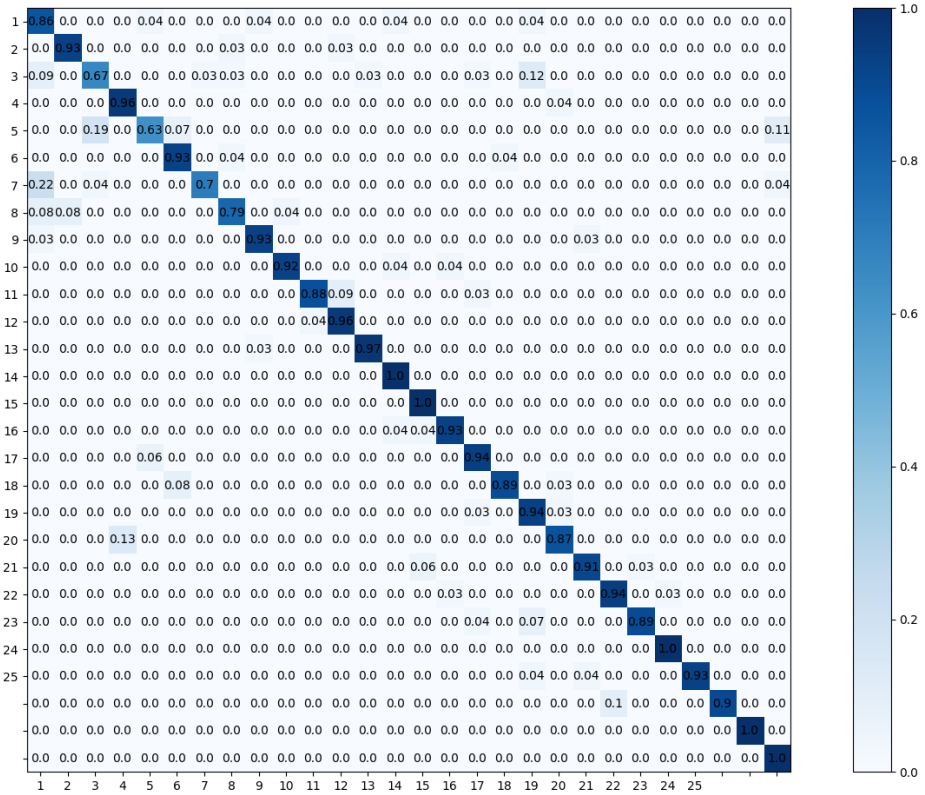


Figure 5.25 – *The confusion matrix obtained using a joint-fine tuning fusion of the RNN_{shape} and the $RNN_{skeleton}$ on the Online DHG dataset for the task of offline recognition of 28 gesture types. Labels of gestures can be found in Table 5.3.*

two handcrafted descriptors based on depth images: the HOG+HOG^2 descriptor proposed by Ohn-bar [105] and the HON4D descriptors proposed by Oreifej *et al.* [107]. Second, we compare our approach to a skeleton-based method proposed by Devanne *et al.* [27] originally designed for human action recognition. Third, we present the results obtained by our approach [23] introduced in Chapter 3. Those methods are introduced in details in Section 3.8.2. Finally, Guerry *et al.* [24] proposed a hand gesture recognition algorithm based on key frames detection followed by a CNN. The recognition accuracies, using both 14 and 28 gesture types of the Online DHG dataset, obtained by the state-of-the-art methods cited below are presented in Table 5.4. We note that the publicly available source codes of these methods are used in our experiments.

Our method outperforms all this state-of-the-art approaches. The key frames detection of *Guerry et al.* [24] leads to a temporal lossy representa-

Table 5.4 – *Comparison with state-of-the-art methods on the Online DHG dataset.*

Method	**14 gestures (%)**	**28 gestures (%)**
Guerry *et al.* [24]	82.90	71.90
De Smedt *et al.* [23]	88.24	81.90
Ohn-Bar *et al.* [105]	83.85	76.53
Oreifej *et al.* [107]	78.53	74.03
Devanne *et al.* [27]	79.61	62.00
Intel RealSense [117] + $RNN_{skeleton}$	84.88	-
Ours, $RNN_{skeleton}$	84.52	76.30
Ours, RNN_{shape}	80.60	76.67
Ours, fusion	**94.17**	**90.48**

tion of the gestures. Beside, the action recognition method of Devanne *et al.* performs poorly and is not suitable for hand gesture recognition.

We note that the $RNN_{skeleton}$ alone does not outperform our hand-crafted approach proposed in [23]. Only by adding the shape features, our framework can outperform this methods by 6%. The effectiveness of the fusion of the hand shape variations and its motions appears truly when looking at results obtained on the task of recognizing 28 gestures, where we outperform state-of-the-art methods by more than 10%.

5.8.3 Online recognition analysis

In this section, we analyze the behavior of our framework on the detection and the recognition processes, as defined in Section 5.7.4, on the Online DHG dataset. We note that in the following experiments, we used the unsegmented sequences of gestures labeled following the vocabulary containing 28 labels.

A) Online detection and recognition

Using the metrics defined in Section 5.7.4, we compute the first ROC curve depicted in Figure 5.26. We choose a gesture detection threshold equal to 0.15 as it shows a good trade-off between a high TPR (85%) and a low FPR (17%).

The NTtD distribution values for various gesture types is shown in Figure 5.27. The average NTtD across all classes is 0.2104, which means that, in average, a hand gesture can be detected after only 21% of its nucleus.

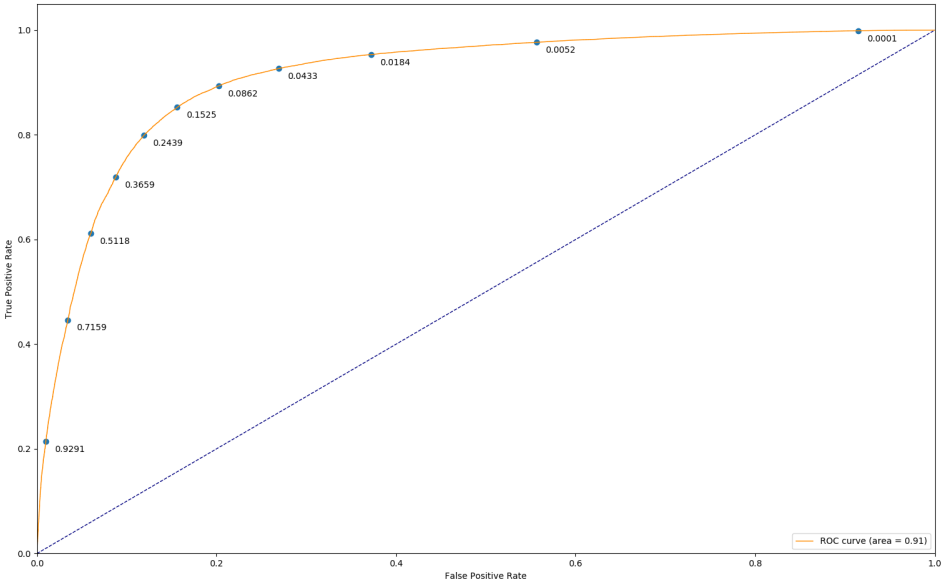


Figure 5.26 – *The ROC curve using our framework on the Online DHG dataset. We obtained an Area Under the Curve equal to* 0.91.

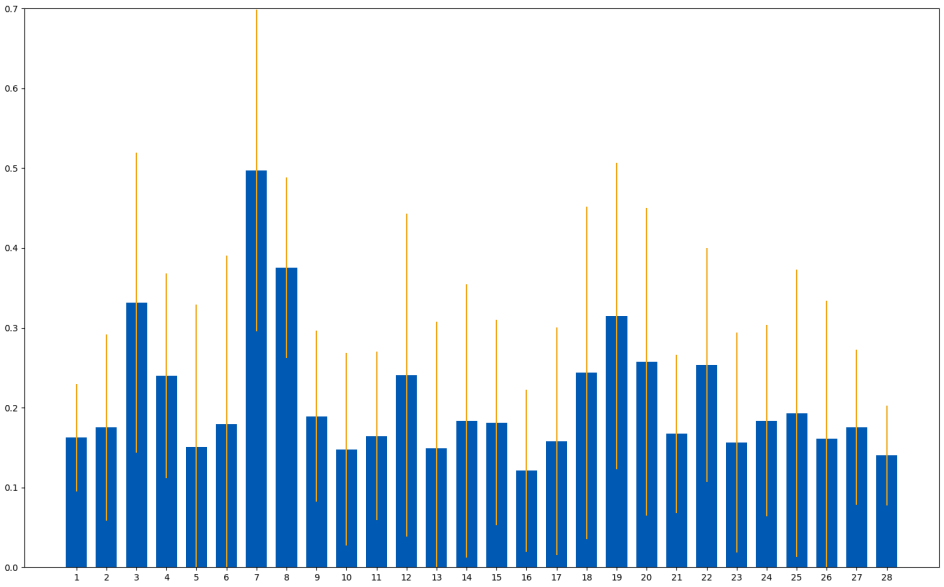


Figure 5.27 – *The Normalized Time to Detect values for the 28 gestures contained in the Online DHG dataset using our framework and a detection threshold ξ equal to* 0.15.

The average nucleus lengths over the whole Online DHG dataset are given in Figure 5.28. We note that nucleus of *fine* gestures (1 - 10) are shorter than those of *coarse* gestures (11 - 25). Moreover, *Swipe* gestures that contain multiple motions, such as *Swipe V*, *X* and + (21 - 28), have naturally the longest nucleus.

Using the detection upstream step, we obtain an overall online accu-

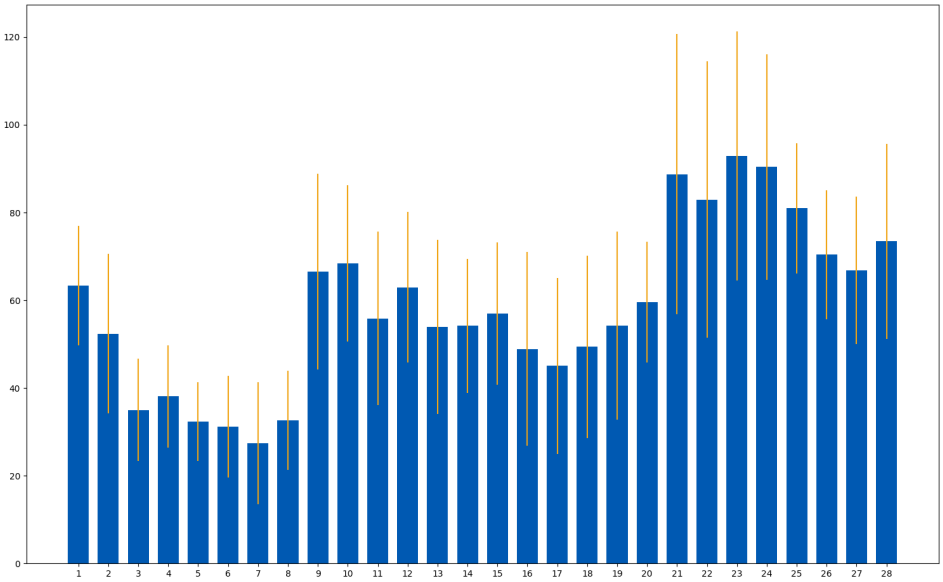


Figure 5.28 – *Nucleus lengths in number of frames of the 28 gestures contained in the Online DHG dataset.*

racy recognition of 82.14%. The confusion matrix is depicted in Figure 5.29.

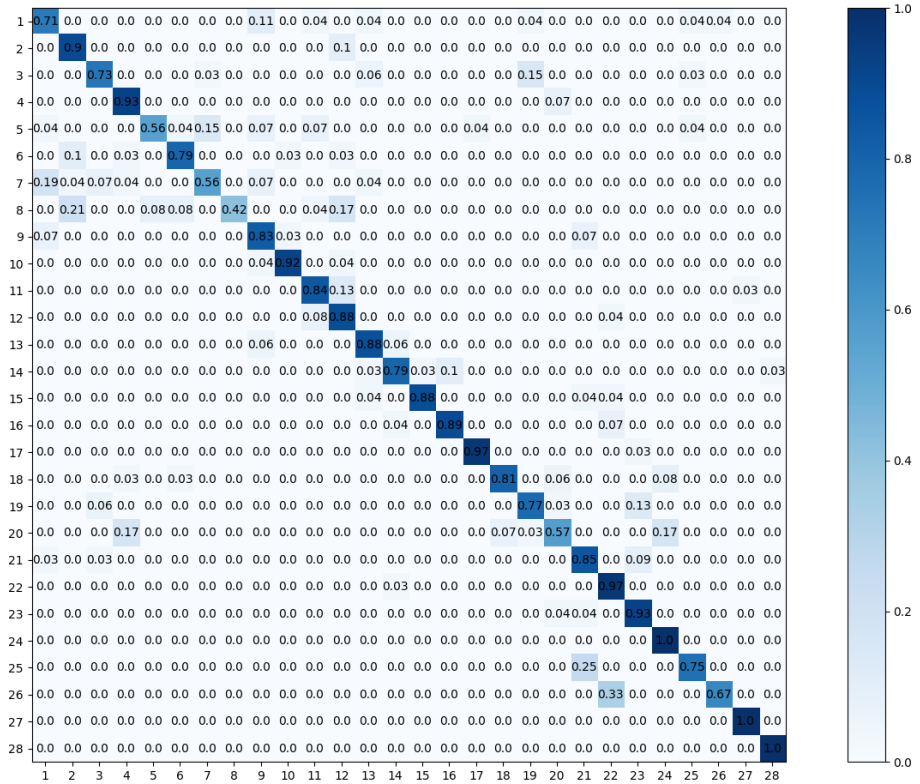


Figure 5.29 – *The confusion matrix obtained on the Online DHG dataset for the task of online recognition of 28 gesture types. Labels of gestures can be found in Table 5.3.*

Due to issues resulting from the detection of gestures, the recognition accuracy decreases by 8.34% compared to the easiest task of pre-segmented gesture recognition. This difference can result from incorrect gesture detection or from confusion between gestures with similar parts. For example, the *Swipe Down* gesture (20) performed with the whole hand obtains an *offline* recognition accuracy up to 87%. In the online scenario, the accuracy decreases by 57% and a high confusion up to 24% appears with the *Swipe V* gesture (24). This can be explained as the first half a *Swipe V* gesture is extremely similar to the *Swipe Down* gesture. In addition, the recognition process suffers from an incorrect prior gesture detection.

B) Discussion

In this section, we analyzed the potential of our framework to perform the detection and the recognition of hand gestures from heterogeneous sets of gesture types. Figure 5.30 illustrates the output of our framework on a test sequence. In this case, the result is almost perfect, each of the 10 gestures is correctly labeled after only few frames.

However, our framework still can be improved. Figure 5.31 shows a test sequence where 5 out of 10 gestures show a missclassification during the first few frames.

Those mistakes arise when our framework detect that a gesture is happening but do not have enough frames, so, not enough information to be recognize correctly. We could overcome issues resulting from those missclassifications by firing an incoming gesture only if its length is longer than a threshold.

5.9 Conclusion

In this chapter, we proposed an effective framework for online hand gesture detection and recognition. Based on statements from a previous work, our goal was to use a transfer learning strategy to perform hand gesture recognition. A CNN trained on a hand pose estimation dataset is used in order to extract hand posture and shape features. Thus, hand gestures represented as depth image sequences were transformed into two distinct

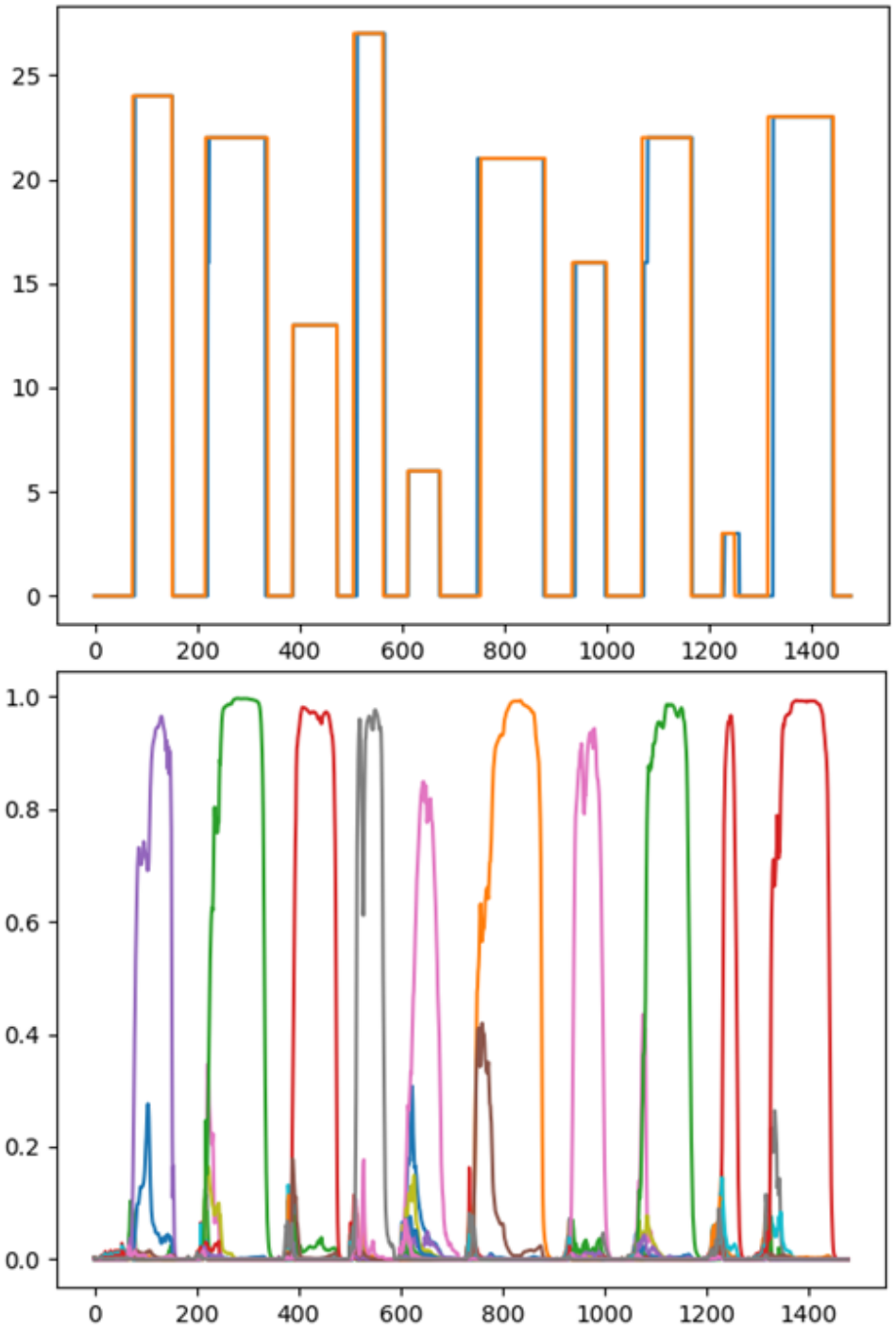


Figure 5.30 – *The gesture detection and recognition performance of our framework on a continuous video stream of 10 gestures. The top figure illustrates the classification outputs of our framework (blue) versus the ground-truth (orange) where the x-axis is the time in number of frames and the y-axis represents the class outputs. The bottom figure represents the accuracy for each gesture types along the time where various colors indicate different gesture types. The no_gesture class is not shown. This figure illustrates an almost perfect recognition result.*

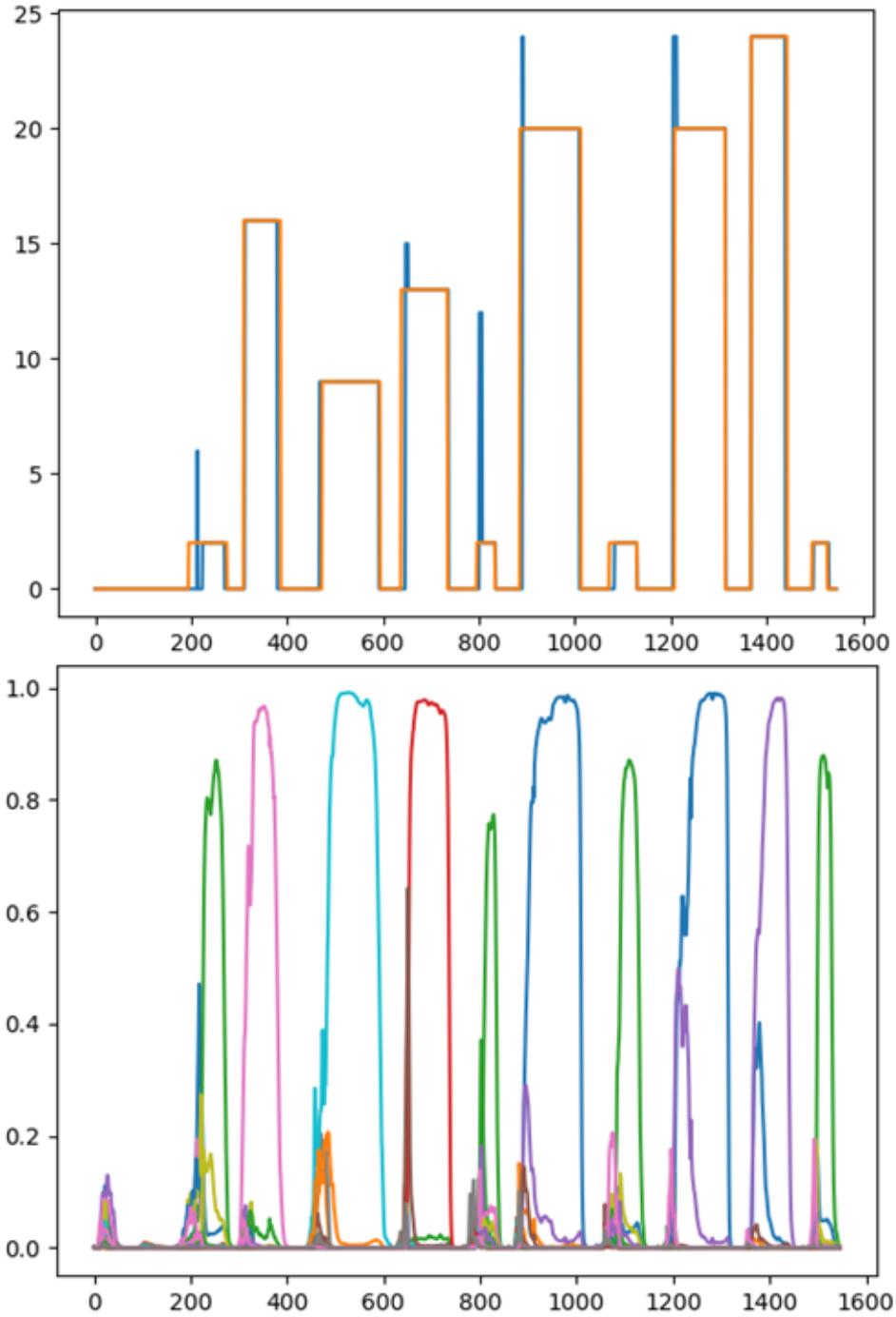


Figure 5.31 – *The gesture detection and recognition performance of our framework on a continuous video stream of 10 gestures. The top figure illustrates the classification outputs of our framework (blue) versus the ground-truth (orange) where the x-axis is the time in number of frames and the y-axis represents the class outputs. The bottom figure represents the accuracy for each gesture types along the time where various colors indicate different gesture types. The no_gesture class is not shown. The top curve shows that five gestures are miss-classified during the first frames of their nucleus.*

ordered lists of hand shapes and hand skeletal features. Then, we used two specific recurrent networks to model separately the temporal aspect of hand shape and skeletal variations through hand gestures. Finally, we fused outputs of both networks to obtain a single class-conditional probability vector by gesture and choose the maximum as the predicted label.

Experiments on the NVIDIA dataset [94] demonstrated that the proposed approach is capable to recognize hand gestures and to improve results of state-of-the-art methods, both, using handcrafted and learned features. Results showed that adding hand shape variation features into the recognition process enhances significantly the overall accuracy for precise hand gesture classification.

For online recognition analysis, we computed the Normalized Time to Detect [51] metric. Results showed that our framework is able to detect an occurring gesture after only 22% of the nucleus phase.

We compared the total number of parameters in our framework (32,555,342) to those of the R3DCNN [94] (79,116,288). The use of a transfer learning strategy allowed us to outperform their result by 1% using less than half parameters.

However, we did not exceed human performance and gestures which contain high hand shape similarities still showed confusions. Thus, hand shape and skeletal features need to be more investigated. Hand pose estimation task, which need also to be improved, is an important challenge for precise and fine hand gestures recognition.

In a second phase, we introduced the Online Dynamic Hand Gesture dataset. We created this dataset in order to study the detection and the recognition of hand gestures from heterogeneous sets of gesture types.

The potential of the fusion of learned features from hand shape variations and its motions appears truly when looking at the results obtained on the task of recognizing 14 gestures types performed with different number of fingers, where we outperform state-of-the-art methods by more than 10%.

Online detection and recognition evaluation results on this challenging dataset show that we can detect an arising gesture after only 21% of

its nucleus. However, our framework shows some missclassification during the first few frames of gestures where the algorithm has been able to detect a gesture in progress but does not yet have sufficient information to correctly recognize its type.

Conclusion

6

Contents

6.1 Summary

In this thesis, we addressed several issues of dynamic hand gesture recognition from depth data, a widely investigated topic due to its wide range of potential applications.

In a first time, we aimed to perform dynamic hand gesture recognition on a set of heterogeneous gesture types. Indeed, the hand is an object with a complex topology. The hand has endless possibilities to perform the same gesture. For example, Feix *et al.* [35] summarized the grasping taxonomy and found 17 distinct hand shapes to perform a grasp. In addition, other gestures, such as *swipes* which are defined by hand motions, are already commonly used in tactile HCI. Thus both hand shape and its motions variations along a gesture have to be taken into account to perform a full recognition process.

Beside, in the field of action recognition, Shotton *et al.* [134] proposed a real-time system to extract 3D positions of a humanoid skeleton from a depth image. Hence, several descriptors in the literature proved how the positions, the motions, and the orientations of skeleton joints could be relevant descriptors for human action recognition. Following those statements, we aimed to perform dynamic hand gesture recognition using hand skeletal features.

As the use of skeletal features for hand gesture recognition is at its beginning, there were no publicly released dataset providing these kind of data. Consequently, we created the Dynamic Hand Gesture dataset providing both hand skeletal sequences in addition to depth images. It contains 2800 samples of 14 gesture types. In order to study the challenge of recognition from an heterogeneous set of gestures, participants were asked to perform the gestures in two ways: using one finger and the whole hand.

We proposed a method to perform dynamic hand gesture recognition using three gestural features computed from hand skeletal sequences: hand direction, rotation and a hand shape descriptor called *Shape of Connected Joint*. Each set of features was encoded in a statistical representation using a Fisher Kernel. In addition, we used a temporal pyramid to model

the dynamic aspect of gestures and a linear SVM to perform the classification step. The evaluation of our approach showed a promising potential of the use of skeletal features to perform hand gesture recognition. Evaluation results demonstrated the efficiency of our approach over the depth image based descriptors. However, the results also revealed a lack of precision to describe the dynamic of complex hand gestures, compared with the feature learning power of modern deep learning approaches.

Recently, deep neural networks have proven their outstanding effectiveness of many area of research. Deep learning became a hot topic since 2010 thanks to the digitization of the society that exponentially increased the size of dataset. Indeed, deep learning algorithms are data-hungry and, while people uploads an increasing amount of data (i.e. texts, photos and videos) every year, the Computer Vision community can use these data to improve the effectiveness of their systems. However, no naturally digitized data, such as hand gesture sequences of depth images, is hard to capture. To be able to take benefit of the power of deep learning algorithms with a smaller amount of data, we can use a smart transfer learning strategy.

In a second time, we aimed to go further in the hand gesture analysis. First, we proposed to perform online recognition which allows the system to detect the presence of a gesture in an unsegmented video stream and to recognize the type of the gesture before its end, which is an essential capacity for real applications. Second, we have taken over the whole pipeline of the recognition process, from hand pose estimation to the classification step, and used the power of deep learning models to increase the efficiency and the robustness of our system.

Our framework is mainly composed of three steps. First, we used a deep learning model which can take image as input – called a Convolutional Neural Network (CNN) – to extract both hand posture and shape descriptors. The CNN is trained using a large hand pose estimation dataset and we transferred its knowledge to extract relevant features. Thus, hand gestures originally represented as depth image sequences are encoded into two distinct lists of features: hand skeletal sequences, which described the hand posture along the gesture, and hand shape feature se-

quences. Second, we used two recurrent networks, designed to take as input time-series data, to model separately the temporal variations of hand postures and its shapes. Finally, we merged outputs of both networks to obtain a single label by gesture. To perform hand gesture detection, we added a *garbage* class. Specifically, we assign the label *garbage* to frames that do not belong to a nucleus phase, i.e. where the gesture occurs.

Experimental results demonstrated that the proposed approach is capable to recognize hand gestures and to improve state-of-the-art results. In addition, the experiments showed that our framework is able to detect the presence of a gesture and to recognize it long before its end, making our system efficient for real-time applications. The use of the transfer learning strategy allowed us to outperform state-of-the-art deep learning approaches using less than half of the number of parameters of the baseline model. However, we did not exceed the human performance and gestures containing high similarities still shows some confusion.

6.2 Future works

Experiments showed that the proposed solutions guarantee an effective dynamic hand gesture recognition but still not exceed the human performance. First, hand pose estimation is still an active field of research and the model designed to extract features used in our framework can be enhanced. In 2017, Yuan *et al.* [172] introduced the million-scale *BigHand2.2M* hand pose dataset which should help researchers to improve the effectiveness of their hand pose estimation algorithms.

Recently, several LSTM configurations have been studied for action recognition application to perform human behavior analysis, such as the hierarchical LSTM [31] or the Spatio-Temporal LSTM [79]. As a new field of study, hand gesture recognition using temporal learned features on skeletal sequences has been only partially studied. Going deeper into the temporal modeling using recurrent layers could provide new ways to distinguish gestures with high similarities. In addition, new recurrent layers have appeared recently providing an attention system. Such networks are able to selectively focus on the informative joint skeletons along a se-

quence. For example, the system could automatically detect the fingers which provides the most reliable information and focus on them to perform the gesture recognition.

In this thesis, we focused on abstract hand gestures (i.e. each gesture has a specific meaning for the system). Meanwhile, in 2017, Garcia *et al.* [38] introduced a hand daily life activities dataset providing sequences of depth images and accurate hand poses. They obtained the best performance using a recurrent neural network on the hand skeletal features. They extracted the hand pose data from a kind of data-glove, since state-of-the-art methods in terms of hand pose estimation still face several issues with fast moving hands or finger self-occlusions. However, the use of a data-glove in real applications is not suitable. An interesting idea would be to use these skeleton data (extracted from the data-glove) during the training phase to support the deep learning model in focusing on relevant features from depth images. Once the model is trained, the skeleton data are not needed anymore for the testing phase. This process is called *regularization*.

The use of a gesture recognition system as the interface with a virtual world can improve the quality of the interaction with the computer. If we focused specifically in this thesis on hand visual cues, other parts of the body can be use as the gaze, the face, the arms, etc. Moreover, in addition to the visual cues, the voice is also an intuitive way to interact with a computer. A system that can process multi-modal data would be able to understand the will of a user with more precision.

Also, using a finite set of abstract gestures to interact with a virtual world is very restrictive. If the systems we proposed in this thesis are able to recognized that a user is grasping a virtual object, it do not have precise information about the required transformation to apply on the object to simulate the real world. To be able to fully reproduce the interaction in the virtual world between the object and the hand, a lot of physical rules has to be taken into account.

Finally, current gesture recognition systems are intrinsically indirect interaction systems and, so, can seem unnatural to users. The barrier of

bringing the sense of touch into a virtual world is a current problematic with many challenges. We can imagine futuristic applications where the limit between the real world and a virtual one is blurring. With this in mind, we still have to develop algorithms able to reproduce the physical rules that guide the interaction between objects and haptic interfaces which can reproduce the sense of touch.

Bibliography

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016. (Cited page 95.)

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012. (Cited page 34.)

[3] ASUS Xtion PRO LIVE. `http://www.asus.com/Multimedia/Xtion_PRO/`, 2013. (Cited page 15.)

[4] J Bakalar. Sony playstation vr review. *CNET, San Francisco, CA, accessed July*, 15:2016, 2016. (Cited page 13.)

[5] Jiatong Bao, Aiguo Song, Yan Guo, and Hongru Tang. Dynamic hand gesture recognition based on surf tracking. In *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*, pages 338–341. IEEE, 2011. (Cited page 30.)

[6] Neha Baranwal and GC Nandi. An efficient gesture based humanoid learning using wavelet descriptor and mfcc techniques. *International Journal of Machine Learning and Cybernetics*, 8(4):1369–1388, 2017. (Cited page 40.)

[7] Mathieu Barnachon, Saïda Bouakaz, Boubakeur Boufama, and Erwan Guillou. Ongoing human action recognition with motion capture. *Pattern Recognition*, 47(1):238–247, 2014. (Cited page 11.)

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up

robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006. (Cited page 30.)

[9] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. (Cited page 91.)

[10] Yoshua Bengio and Yann LeCun. NIPS 2015 Deep Learning Tutorial. (Cited page 108.)

[11] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997. (Cited page 146.)

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. (Cited pages 39, 40, and 92.)

[13] Lars Bretzner, Ivan Laptev, and Tony Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pages 423–428. IEEE, 2002. (Cited page 21.)

[14] Ian M Bullock, Thomas Feix, and Aaron M Dollar. The yale human grasping dataset: Grasp, object, and task data in household and machine shop environments. *The International Journal of Robotics Research*, 34(3):251–255, 2015. (Cited pages 27 and 28.)

[15] Minjie Cai, Kris M Kitani, and Yoichi Sato. A scalable approach for understanding the visual structures of hand grasps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1360–1366. IEEE, 2015. (Cited pages 27 and 28.)

[16] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011. (Cited page 66.)

[17] Feng-Sheng Chen, Chih-Ming Fu, and Chung-Lin Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 21(8):745–758, 2003. (Cited page 39.)

[18] Hong Cheng, Zhongjun Dai, and Zicheng Liu. Image-to-class dynamic time warping for 3d hand gesture recognition. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013. (Cited pages 27, 28, 33, and 51.)

[19] Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Ng Andrew. Deep learning with cots hpc systems. In *International Conference on Machine Learning*, pages 1337–1345, 2013. (Cited page 96.)

[20] Simon Conseil, Salah Bourennane, and Lionel Martin. Comparison of fourier descriptors and hu moments for hand posture recognition. In *Signal Processing Conference, 2007 15th European*, pages 1960–1964. IEEE, 2007. (Cited page 33.)

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. (Cited pages 30 and 37.)

[22] Nasser H Dardas and Nicolas D Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3592–3607, 2011. (Cited pages 11, 31, and 40.)

[23] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeborre. Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2016. (Cited pages 56, 120, 124, 158, and 159.)

[24] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, Joris Guerry, Bertrand Le Saux, and David Filliat. Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In *10th*

Eurographics Workshop on 3D Object Retrieval, 2017. (Cited pages 158 and 159.)

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. (Cited pages 95 and 120.)

[26] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. Hand3d: Hand pose estimation using 3d neural network. *arXiv preprint arXiv:1704.02224*, 2017. (Cited page 24.)

[27] Maxime Devanne, Hazem Wannous, Stefano Berretti, Pietro Pala, Mohamed Daoudi, and Alberto Del Bimbo. 3-d human action recognition by shape analysis of motion trajectories on riemannian manifold. *IEEE transactions on cybernetics*, 45(7):1340–1352, 2015. (Cited pages 16, 36, 75, 76, 77, 88, 158, and 159.)

[28] Fabio Dominio, Mauro Donadeo, and Pietro Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. *Pattern Recognition Letters*, 50:101–111, 2014. (Cited pages 35 and 36.)

[29] Cao Dong, Ming C Leu, and Zhaozheng Yin. American sign language alphabet recognition using microsoft kinect. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 44–52, 2015. (Cited page 35.)

[30] Philippe Dreuw, Carol Neidle, Vassilis Athitsos, Stan Sclaroff, and Hermann Ney. Benchmark databases for video-based automatic sign language recognition. In *LREC*, 2008. (Cited page 14.)

[31] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015. (Cited pages 43, 44, and 170.)

[32] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. (Cited pages xiii and 110.)

[33] Sergio Escalera, Xavier Baró, Jordi Gonzalez, Miguel Ángel Bautista, Meysam Madadi, Miguel Reyes, Víctor Ponce-López, Hugo Jair Escalante, Jamie Shotton, and Isabelle Guyon. Chalearn looking at people challenge 2014: Dataset and results. In *ECCV Workshops (1)*, pages 459–473, 2014. (Cited pages 27, 28, 51, and 143.)

[34] Georgios Evangelidis, Gurkirt Singh, and Radu Horaud. Skeletal quads: Human action recognition using joint quadruples. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4513–4518. IEEE, 2014. (Cited pages 63, 64, 76, and 77.)

[35] Thomas Feix, Roland Pawlik, Heinz-Bodo Schmiedmayer, Javier Romero, and Danica Kragic. A comprehensive grasp taxonomy. In *Robotics, science and systems: workshop on understanding the human hand for advancing robotic manipulation*, volume 2, pages 2–3, 2009. (Cited pages 26, 51, and 168.)

[36] Bin Feng, Fangzi He, Xinggang Wang, Yongjiang Wu, Hao Wang, Sihua Yi, and Wenyu Liu. Depth-projection-map-based bag of contour fragments for robust hand gesture recognition. *IEEE Transactions on Human-Machine Systems*, 47(4):511–523, 2017. (Cited pages 27 and 28.)

[37] Simon Fothergill, Helena Mentis, Pushmeet Kohli, and Sebastian Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1737–1746. ACM, 2012. (Cited page 57.)

[38] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. *arXiv preprint arXiv:1704.02463*, 2017. (Cited pages 27, 28, 36, 43, and 171.)

[39] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. Vision based hand gesture recognition. *World Academy of Science, Engineering and Technology*, 49(1):972–977, 2009. (Cited page 21.)

[40] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3593–3601, 2016. (Cited pages 17, 25, 79, 81, 82, 84, and 123.)

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. (Cited pages 91, 101, and 113.)

[42] Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012. (Cited page 150.)

[43] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006. (Cited pages 41 and 150.)

[44] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. Region ensemble network: Improving convolutional network for hand pose estimation. *arXiv preprint arXiv:1702.02447*, 2017. (Cited page 24.)

[45] Bhumika Gupta, Pushkar Shukla, and Ankush Mittal. K-nearest correlated neighbor classification for indian sign language gesture recognition using feature fusion. In *Computer Communication and Informatics (ICCCI), 2016 International Conference on*, pages 1–5. IEEE, 2016. (Cited page 40.)

[46] Shalini Gupta, Pavlo Molchanov, Xiaodong Yang, Kihwan Kim, Stephen Tyree, and Jan Kautz. Towards selecting robust hand gestures for automotive interfaces. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 1350–1357. IEEE, 2016. (Cited pages 83 and 134.)

[47] Jungong Han, Ling Shao, Dong Xu, and Jamie Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE transactions on cybernetics*, 43(5):1318–1334, 2013. (Cited page 16.)

[48] Tony Heap and David Hogg. Towards 3d hand tracking using a deformable model. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 140–145. IEEE, 1996. (Cited page 21.)

[49] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998. (Cited pages 39, 40, 66, and 92.)

[50] O Ben Henia, Mohamed Hariti, and Saida Bouakaz. A two-step minimization algorithm for model-based hand tracking. 2010. (Cited page 21.)

[51] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014. (Cited pages 146 and 165.)

[52] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. (Cited page 111.)

[53] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine Vision and Applications*, 27(7):1005–1020, 2016. (Cited page 16.)

[54] Jian-Fang Hu, Wei-Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5344–5352, 2015. (Cited page 36.)

[55] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. (Cited page 105.)

[56] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. (Cited page 91.)

[57] Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2983–2991, 2015. (Cited page 132.)

[58] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017. (Cited page 116.)

[59] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. (Cited pages 41, 43, and 150.)

[60] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011. (Cited page 22.)

[61] Microsoft Kinect. `http://www.microsoftcom/en-us/kinectforwindows`, 2013. (Cited pages 11, 15, 16, and 28.)

[62] Alexander Klaser, Marcin Marszaek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, pages 275–1. British Machine Vision Association, 2008. (Cited page 37.)

[63] Eva Kollorz, Jochen Penne, Joachim Hornegger, and Alexander Barke. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3-4):334–343, 2008. (Cited page 29.)

[64] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques, 2007. (Cited page 39.)

[65] Ivan Krasin, Tom Duerig, Neil Alldrin, Andreas Veit, Sami Abu-El-Haija, Serge Belongie, David Cai, Zheyun Feng, Vittorio Ferrari, Victor Gomes, Abhinav Gupta, Dhyanesh Narayanan, Chen Sun,

Gal Chechik, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016. (Cited page 113.)

[66] Alexey Kurakin, Zhengyou Zhang, and Zicheng Liu. A real time system for dynamic hand gesture recognition with a depth sensor. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1975–1979. IEEE, 2012. (Cited pages 27, 28, 29, 32, 39, and 51.)

[67] Alina Kuznetsova, Laura Leal-Taixé, and Bodo Rosenhahn. Real-time sign language recognition using a consumer depth camera. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 83–90, 2013. (Cited page 51.)

[68] Yann Le Cun, LD Jackel, B Boser, JS Denker, HP Graf, I Guyon, D Henderson, RE Howard, and W Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989. (Cited page 106.)

[69] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited pages 40, 104, and 106.)

[70] Hyeon-Kyu Lee and Jin-Hyung Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 21(10):961–973, 1999. (Cited page 11.)

[71] Jong Lee-Ferng, Javier Ruiz-del Solar, Rodrigo Verschae, and Mauricio Correa. Dynamic gesture recognition for human robot interaction. In *Robotics Symposium (LARS), 2009 6th Latin American*, pages 1–8. IEEE, 2009. (Cited page 39.)

[72] Shao-Zi Li, Bin Yu, Wei Wu, Song-Zhi Su, and Rong-Rong Ji. Feature

learning based on sae–pca network for human gesture recognition in rgbd images. *Neurocomputing*, 151:565–573, 2015. (Cited page 40.)

[73] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Expandable data-driven graphical modeling of human actions based on salient postures. *IEEE transactions on Circuits and Systems for Video Technology*, 18(11):1499–1510, 2008. (Cited page 39.)

[74] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 9–14. IEEE, 2010. (Cited page 56.)

[75] Hui Liang, Junsong Yuan, and Daniel Thalmann. Parsing the hand in depth images. *IEEE Transactions on Multimedia*, 16(5):1241–1253, 2014. (Cited pages 79 and 123.)

[76] Hui Liang, Junsong Yuan, and Daniel Thalmann. Resolving ambiguous hand pose predictions by exploiting part correlations. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(7):1125–1139, 2015. (Cited pages 79 and 123.)

[77] Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen. Human hand gesture recognition using a convolution neural network. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1038–1043. IEEE, 2014. (Cited page 40.)

[78] Zhe Lin, Zhuolin Jiang, and Larry S Davis. Recognizing actions by shape-motion prototype trees. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 444–451. IEEE, 2009. (Cited page 27.)

[79] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833. Springer, 2016. (Cited page 170.)

[80] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recog-

nition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 7, 2017. (Cited pages 43 and 44.)

[81] Li Liu and Ling Shao. Learning discriminative representations from rgb-d video data. In *IJCAI*, volume 4, page 8, 2013. (Cited pages 27 and 28.)

[82] Liwei Liu, Junliang Xing, Haizhou Ai, and Xiang Ruan. Hand posture recognition using finger geometric feature. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 565–568. IEEE, 2012. (Cited page 40.)

[83] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. (Cited page 30.)

[84] Wei Lu, Zheng Tong, and Jinghui Chu. Dynamic hand gesture recognition with leap motion controller. *IEEE Signal Processing Letters*, 23(9):1188–1192, 2016. (Cited pages 27, 28, 36, 43, and 74.)

[85] Dan MacIsaac et al. Google cardboard: A virtual reality headset for 10? *The Physics Teacher*, 53(2):125–125, 2015. (Cited page 13.)

[86] Meysam Madadi, Sergio Escalera, Xavier Baro, and Jordi Gonzalez. End-to-end global to local cnn learning for hand pose recovery in depth data. *arXiv preprint arXiv:1705.09606*, 2017. (Cited page 24.)

[87] Sotiris Manitsaris, Apostolos Tsagaris, Alina Glushkova, Fabien Moutarde, and Frédéric Bevilacqua. Fingers gestures early-recognition with a unified framework for rgb or depth camera. In *Proceedings of the 3rd International Symposium on Movement and Computing*, page 26. ACM, 2016. (Cited page 39.)

[88] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 1565–1569. IEEE, 2014. (Cited pages 27, 28, 36, and 40.)

[89] Carlos Mateo Agulló, Pablo Gil, Corrales Ramón, Juan Antonio, Santiago Timoteo Puente Méndez, and Fernando Torres. Rgbd human-hand recognition for the interaction with robot-hand. 2012. (Cited page 34.)

[90] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. (Cited page 94.)

[91] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, pages 63–70. Canadian Information Processing Society, 2013. (Cited page 18.)

[92] Mohammad Moghimi, Pablo Azagra, Luis Montesano, Ana C Murillo, and Serge Belongie. Experiments on an rgb-d wearable vision system for egocentric activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 597–603, 2014. (Cited pages 27 and 28.)

[93] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–7, 2015. (Cited pages 41, 42, and 51.)

[94] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016. (Cited pages 27, 28, 41, 42, 51, 82, 84, 85, 128, 133, 134, 150, 151, and 165.)

[95] Camille Monnier, Stan German, and Andrey Ost. A multi-scale boosted detector for efficient and robust gesture recognition. In *ECCV Workshops (1)*, pages 491–502, 2014. (Cited page 51.)

[96] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber,

and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347. IEEE, 2011. (Cited page 40.)

[97] Natalia Neverova, Christian Wolf, Florian Nebout, and Graham Taylor. Hand pose estimation through semi-supervised and weakly-supervised learning. *arXiv preprint arXiv:1511.06728*, 2015. (Cited page 24.)

[98] Natalia Neverova, Christian Wolf, Graham Taylor, and Florian Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016. (Cited pages 41 and 143.)

[99] Natalia Neverova, Christian Wolf, Graham W Taylor, and Florian Nebout. Multi-scale deep learning for gesture detection and localization. In *Workshop at the European conference on computer vision*, pages 474–490. Springer, 2014. (Cited page 88.)

[100] Markus Oberweger, Gernot Riegler, Paul Wohlhart, and Vincent Lepetit. Efficiently creating 3d training data for fine hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4957–4965, 2016. (Cited pages 18 and 19.)

[101] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015. (Cited pages 17, 24, 30, 79, 81, 82, 84, 123, and 125.)

[102] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3316–3324, 2015. (Cited pages 17 and 123.)

[103] VR Oculus. Oculus rift. *Available from WWW: http://www.oculusvr.com/rift*, 2015. (Cited page 13.)

[104] Eshed Ohn-Bar and Mohan Trivedi. Joint angles similarities and hog2 for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 465–470, 2013. (Cited pages 37, 75, 76, and 77.)

[105] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014. (Cited pages 27, 28, 37, 51, 57, 84, 150, 158, and 159.)

[106] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011. (Cited page 22.)

[107] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2013. (Cited pages 75, 76, 77, 158, and 159.)

[108] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. (Cited page 114.)

[109] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *European Conference on Computer Vision*, pages 581–595. Springer, 2014. (Cited page 63.)

[110] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012. (Cited page 27.)

[111] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The leap motion controller: a view on sign language. *Australian computer-human interaction conference: augmentation, application, innovation, collaboration*, pages 175–178, 2013. (Cited pages 36 and 52.)

[112] Lily Prasuethsut. Htc vive: Everything you need to know about the steamvr headset. *Retrieved January*, 3:2017, 2016. (Cited page 13.)

[113] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125, 1994. (Cited page 68.)

[114] Nicolas Pugeault and Richard Bowden. Spelling it out: Real-time asl fingerspelling recognition. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1114–1119. IEEE, 2011. (Cited pages 27, 28, 30, 40, and 51.)

[115] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Real-time and robust hand tracking from depth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, 2014. (Cited pages 18 and 19.)

[116] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986. (Cited page 92.)

[117] Intel RealSense. `http://www.intel.com/realsense`, 2016. (Cited pages 11, 16, 17, 58, 81, 82, 152, 154, and 159.)

[118] J Rekha, J Bhattacharya, and S Majumder. Shape, texture and local movement hand gesture features for indian sign language recognition. In *Trendz in Information Sciences and Computing (TISC), 2011 3rd International Conference on*, pages 30–35. IEEE, 2011. (Cited page 40.)

[119] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia*, 15(5):1110–1120, 2013. (Cited pages 27, 28, 33, 34, and 51.)

[120] Grégory Rogez, James S Supancic, and Deva Ramanan. First-person pose recognition using egocentric workspaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4325–4333, 2015. (Cited pages 18 and 19.)

[121] Grégory Rogez, James S Supancic, and Deva Ramanan. Understanding everyday hands in action from rgb-d images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3889–3897, 2015. (Cited pages 27 and 28.)

[122] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. (Cited page 94.)

[123] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. 1985. (Cited pages 99 and 109.)

[124] Samsung Gear VR. `http://www.samsung.com/fr/consumer/mobile-devices/smartphones/galaxy-s/galaxy-s7/gear-vr/`, 2016. (Cited page 13.)

[125] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013. (Cited pages 63 and 64.)

[126] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. (Cited page 96.)

[127] Stephen J Schmugge, Sriram Jayaram, Min C Shin, and Leonid V Tsap. Objective evaluation of approaches of skin detection using roc analysis. *Computer Vision and Image Understanding*, 108(1):41–51, 2007. (Cited page 29.)

[128] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. (Cited page 43.)

[129] Lorenzo Seidenari, Vincenzo Varano, Stefano Berretti, Alberto Bimbo, and Pietro Pala. Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 479–485, 2013. (Cited page 56.)

[130] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1010–1019, 2016. (Cited page 56.)

[131] ShapeHand. `http://http://www.shapehand.com/shapehand.html`, 2009. (Cited pages 18 and 20.)

[132] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3633–3642. ACM, 2015. (Cited pages 19, 20, and 57.)

[133] Nobutaka Shimada, Yoshiaki Shirai, Yoshinori Kuno, and Jun Miura. Hand gesture estimation and model refinement using monocular camera-ambiguity limitation by inequality constraints. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 268–273. IEEE, 1998. (Cited page 33.)

[134] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013. (Cited pages 16, 30, 35, 51, 52, 56, and 168.)

[135] Softkinetic. `https://www.softkinetic.com`, 2013. (Cited page 17.)

[136] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2456–2463, 2013. (Cited pages 18 and 19.)

[137] Ekaterini Stergiopoulou and Nikos Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *En-*

gineering Applications of Artificial Intelligence, 22(8):1141–1158, 2009. (Cited page 34.)

[138] Erik B Sudderth, Michael I Mandel, William T Freeman, and Alan S Willsky. Visual hand tracking using nonparametric belief propagation. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 189–189. IEEE, 2004. (Cited page 33.)

[139] Heung-Il Suk, Bong-Kee Sin, and Seong-Whan Lee. Hand gesture recognition based on dynamic bayesian network framework. *Pattern recognition*, 43(9):3059–3072, 2010. (Cited page 11.)

[140] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 824–832, 2015. (Cited pages 18, 19, 23, 79, 81, and 123.)

[141] James Steven Supancic III, Gregory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. Depth based hand pose estimation: methods, data, and challenges. arxiv preprint. *arXiv preprint arXiv:1504.06378*, 2015. (Cited page 29.)

[142] Poonam Suryanarayan, Anbumani Subramanian, and Dinesh Mandalapu. Dynamic hand pose recognition using depth data. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 3105–3108. IEEE, 2010. (Cited page 29.)

[143] Hironori Takimoto, Jaemin Lee, and Akihiro Kanagawa. A robust gesture recognition using depth data. *International Journal of Machine Learning and Computing*, 3(2):245, 2013. (Cited pages 29 and 31.)

[144] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3786–3793, 2014. (Cited pages 18, 19, 20, 23, 57, 81, 121, 124, 125, and 134.)

[145] Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3325–3333, 2015. (Cited page 23.)

[146] Alaa Tharwat, Tarek Gaber, Aboul Ella Hassanien, MK Shahin, and Basma Refaat. Sift-based arabic sign language recognition system. In *Afro-european conference for industrial advancement*, pages 359–370. Springer, 2015. (Cited page 40.)

[147] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):169, 2014. (Cited pages 17, 18, 19, 20, 30, and 124.)

[148] NDI trakSTAR. `https://www.ascension-tech.com/products/trakstar-2-drivebay-2/`. (Cited page 20.)

[149] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. (Cited pages 84, 85, and 150.)

[150] Michael Van den Bergh and Luc Van Gool. Combining rgb and tof cameras for real-time 3d hand gesture interaction. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 66–72. IEEE, 2011. (Cited page 30.)

[151] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014. (Cited pages 16 and 36.)

[152] Chong Wang, Zhong Liu, and Shing-Chow Chan. Superpixel-based hand gesture recognition with kinect depth camera. *IEEE transactions on multimedia*, 17(1):29–39, 2015. (Cited pages 27, 28, 34, 35, and 51.)

[153] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. (Cited page 30.)

[154] Hanjie Wang, Qi Wang, and Xilin Chen. Hand posture recognition from disparity cost map. In *Asian Conference on Computer Vision*, pages 722–733. Springer, 2012. (Cited page 51.)

[155] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013. (Cited pages 36 and 63.)

[156] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. *arXiv preprint arXiv:1704.02581*, 2017. (Cited page 43.)

[157] Jiang Wang, Zicheng Liu, and Ying Wu. Learning actionlet ensemble for 3d human action recognition. In *Human Action Recognition with Depth Cameras*, pages 11–40. Springer, 2014. (Cited page 16.)

[158] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1290–1297. IEEE, 2012. (Cited page 56.)

[159] Sy Bor Wang, Ariadna Quattoni, L-P Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1521–1527. IEEE, 2006. (Cited page 39.)

[160] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. (Cited page 130.)

[161] Aaron Wetzler, Ron Slossberg, and Ron Kimmel. Rule of thumb:

Deep derotation for improved fingertip detection. *arXiv preprint arXiv:1507.05726*, 2015. (Cited pages 18, 19, and 20.)

[162] Di Wu, Lionel Pigou, Pieter-Jan Kindermans, Nam Do-Hoang Le, Ling Shao, Joni Dambre, and Jean-Marc Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1583–1597, 2016. (Cited pages 137 and 139.)

[163] Ying Wu, John Y Lin, and Thomas S Huang. Capturing natural hand articulation. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 426–432. IEEE, 2001. (Cited page 126.)

[164] Chi Xu, Ashwin Nanjappa, Xiaowei Zhang, and Li Cheng. Estimate hand poses efficiently from single depth images. *International Journal of Computer Vision*, 116(1):21–45, 2016. (Cited pages 18 and 19.)

[165] Yuanrong Xu, Qianqian Wang, Xiao Bai, Yen-Lun Chen, and Xinyu Wu. A novel feature extracting method for dynamic gesture recognition based on support vector machine. In *Information and Automation (ICIA), 2014 IEEE International Conference on*, pages 437–441. IEEE, 2014. (Cited pages 27 and 28.)

[166] Hongwei Yang and Juyong Zhang. Hand pose regression via a classification-guided approach. In *Asian Conference on Computer Vision*, pages 452–466. Springer, 2016. (Cited page 23.)

[167] Xiaodong Yang and YingLi Tian. Super normal vector for activity recognition using depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 804–811, 2014. (Cited pages 84 and 150.)

[168] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. Recognizing actions using depth motion maps-based histograms of oriented gradients. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1057–1060. ACM, 2012. (Cited page 37.)

[169] Yi Yao and Chang-Tsun Li. Hand posture recognition using surf with adaptive boosting. In *Electronic Proceedings of the British Machine Vision Conference*, pages 1–10. British Machine Vision Association and Society for Pattern Recognition, 2012. (Cited page 30.)

[170] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *European Conference on Computer Vision*, pages 346–361. Springer, 2016. (Cited pages 17 and 24.)

[171] Ho-Sub Yoon, Jung Soh, Younglae J Bae, and Hyun Seung Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern recognition*, 34(7):1491–1501, 2001. (Cited page 11.)

[172] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis. *arXiv preprint arXiv:1704.02612*, 2017. (Cited pages 19, 20, 45, and 170.)

[173] Gloria Zen, Lorenzo Porzi, Enver Sangineto, Elisa Ricci, and Nicu Sebe. Learning personalized models for facial expression analysis and gesture recognition. *IEEE Transactions on Multimedia*, 18(4):775–788, 2016. (Cited page 63.)

[174] Chenyang Zhang and Yingli Tian. Edge enhanced depth motion map for dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 500–505, 2013. (Cited pages 37, 38, and 40.)

[175] Chenyang Zhang, Xiaodong Yang, and YingLi Tian. Histogram of 3d facets: A characteristic descriptor for hand gesture recognition. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, pages 1–8. IEEE, 2013. (Cited pages 31, 40, 51, and 64.)

[176] Hanning Zhou et al. Tracking articulated hand motion with eigen dynamics analysis. In *Computer Vision, 2003. Proceedings. Ninth IEEE*

International Conference on, pages 1102–1109. IEEE, 2003. (Cited page 21.)

[177] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. Model-based deep hand pose estimation. *arXiv preprint arXiv:1606.06854*, 2016. (Cited pages 17 and 24.)