HAND GESTURE RECOGNITION FOR SIGN LANGUAGE TRANSCRIPTION

by

Iker Vazquez Lopez

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

May 2017

© 2017
Iker Vazquez Lopez
ALL RIGHTS RESERVED

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Iker Vazquez Lopez

Thesis Title: Hand Gesture Recognition for Sign Language Transcription

Date of Final Oral Examination: 15 March 2017

The following individuals read and discussed the thesis submitted by student Iker Vazquez Lopez, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Steven M. Cutchin, Ph.D.	Co-Chair, Supervisory Committee
Jerry Alan Fails, Ph.D.	Co-Chair, Supervisory Committee
Maria Soledad Pera, Ph.D.	Member, Supervisory Committee
Casey Kennington, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Steve Cutchin, Ph.D., Co-Chair and Jerry Alan Fails, Co-Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

Nire aitonaren oroimenean, etxetik oso urrun nengoela, hegoak hazi eta nire amonarengana hegan joan zena.

ACKNOWLEDGMENTS

Before starting this document I would like to thank my parents for the education and all the support they provided to me along these last years. I would like to thank and mention my family and friends too for asking every time I was with them about how my studies were going and how I was feeling.

In the other hand, I would like to thank the committee members for giving me advice on how to walk the path of developing this thesis and for helping when I was stuck in a problem I could not solve giving another point of view of it.

Finally, I would like to mention all the friends I made since I came to Boise and how they made my life easier taking me out of the working routine. I really think it helped me to not get only focused on my work, so I could rest from becoming very stressed.

ABSTRACT

Sign Language is a language which allows mute people to communicate. The ease of communication offered by this language, however, disappears when one of the interlocutors, who may or not be mute, does not know Sign Language and a conversation starts using that language. In this document, I discuss research efforts that resulted in a system that takes advantage of Convolutional Neural Networks to recognize hand letter and number gestures from American Sign Language based on depth images captured by the Kinect camera. As a byproduct of these research efforts, I created a new dataset which consists of depth images of American Sign Language letters and numbers, I conducted an empirical assessment and I compared the presented method for image recognition against a similar dataset but for Vietnamese Sign Language. Finally, I present how this work supports my ideas for the future work on a complete system for Sign Language transcription.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, the use of different types of controls besides mouse and keyboard has become common. A number of devices are available for specific tasks or applications, one interaction form that has gained popularity is the area of natural interactions between humans and machines. Web browsers, video games, Virtual Reality (VR) environments and a diverse set of tools have taken advantage of users' natural interactions, e.g., voice control, touchpads, haptic devices, cameras, etc. Immersing the user into the systems or environments in a natural way is the goal of this type of research. Natural interaction involves the use of a user's body without additional hardware. This is called Natural User Interface (NUI). By using sensors and capturing the diverse interactions of the user's body it is possible to recognize commands and perform required tasks in a system.

NUI is the use of body and hand gestures, therefore Hand Gesture Recognition (HGR) is an essential component of this kind of system. Smartphones and tablets are controlled by touching the device and performing gestures. Another method for NUIs is the use of cameras to detect the user's different body parts like the Xbox Kinect camera does. The hands of the user can provide a collection of gestures and if the gesture is recognized, allow control of applications. Segmenting the hands from cameras and knowing which gesture is being performed lead scientists to one of the areas in our research community that has been studied for years: Sign Language Recognition (SLR).

Sign Language (SL) is not commonly learned by non-mute people, thus, mute people have problems communicating. Usually, people do not learn it if there is no mute person in their relation circles or if it is not required for their job. When

they engage with a mute person the communication can be hard and tedious. As an example, a mute individual goes to an interview: if the interviewer does not know SL the common approach is to hire a translator. This action creates some problems, as hiring can be expensive and scheduling an appointment with three people, depending on the circumstances, difficult. This is where image recognition techniques play an important role by automatizing the process of identifying signs. The focus of this research is on this type of problem: facilitating communication via SL by automating the transcription of SL without the need for a human translator.

Even though SLR research using machines started a long time ago, because of the challenge of the problem, there is not a full automatic Sign Language transcriptor system. Dealing with the scalability of the problem (number of gestures and variations, movement, face expressions and contextual meaning) is not trivial. This is one reason why previous studies focus on limited datasets of gestures and users. SL has hundreds of symbols to represent many concepts. These symbols have variations that change the meaning of them depending on the context and what the performer wants to say: pointing, arm moving and face expressions are examples of variances. As an example of meaning variation of a sign, consider the phrase *not yet*, the sign for the phrase requires the tongue touching the lower lip and the head rotating from side to side. If both actions are not performed the meaning of the sign becomes *late*.

The goal of this research is to study the performance of a Convolutional Neural Network (CNN) recognizing and transcribing into text SL images (gestures performed by a hand). Given the broad scope of this task I limited the scope of the study to American Sign Language (ASL) letters and number symbols. Since some of the

gestures (J and Z) need some movement to be performed I discarded them. I kept all the gestures that can be performed in a static manner. I defined a scene which has also been predefined in an "ideal world" to capture the images in a consistent way. This avoids errors in the image capture step. From the limited and ideal scenario, adding more gestures and more features (like hand and arm movement, and face expressions) to the system and modifying the environment into a more intricate one will allow the generalization of the problem, resulting in a more robust system.

The research work presented in this document is focused on creating a classification model for hand letter and number gestures and provides a new dataset for hand gesture recognition. The model detects local features of the hand images and recognizes the gesture in each of the images. The created dataset has fixed size images, the hands are centered, and the values of the hands normalized. Since all the preprocessing steps have been done to the images, all the images are in the same coordinates so they can be compared against each other directly.

The rest of this document is organized as follows. In Chapter 2, I present studies of previous works in the area of human body recognition and SL recognition. Since I used a camera to capture the images, I also discuss diverse sensors available that can be used for gathering data (i.e., hand gestures in my case). I discuss the advantages and disadvantages of each one and why I selected a camera which can capture Red, Green and Blue colors together with the Depth information of the scene. I describe the methods for a user's body recognition and some works in which those techniques are applied. In Chapter 3, I list constraints of the SL recognition problem together with the approach I developed and the construction of the CNN. I organized this

chapter into 2 major subsections: I discuss an initial study and preliminary tests to bound the scope of the problem and identify the constraints of it. Based on the initial study results and conclusions, I tested an approach based on a previous work for face recognition and I explain my approach of developing and implementing a classification model using a CNN. In Chapter 4, I discuss the ideal dataset I would like to have and the characteristics of the dataset I captured. I present different evaluation methods used to test the classification model and the results from the captured dataset. Finally, in Chapter 5, I present conclusions of this work and list future work I am planning for developing in Sign Language Recognition.

CHAPTER 2

RELATED WORK

Sign Language Recognition involves a variety of techniques from diverse areas. In this chapter, I present prior work related to Sign Language Recognition. The first step of Sign Language Recognition is to capture the symbols performed by the user. To do this I examine sensors which provide a best balance between frame-rate, accuracy, and affordability. I present a set of sensors which cover a raise of capture devices and their aspects. With sensors to capture the scene the next step is to track and recognize the user's body and different parts of it, such as face, arms, and hands.

Methods for recognizing bodies and body parts have been widely studied and a diverse set of applications created. Neural Networks (NN) are often used as recognition models for image processing [25]. I present a basic NN model and some variations applied to different areas. I focus on a specific type of network, Convolutional Neural Network (CNN), a popular NN in image processing because of its success recognizing local features. The recognized features are often used as input to NUI systems when interaction is via a user's body. This way of interacting avoids the use of physical control devices, allowing natural communication with the computer. The focus of NUI is to develop methods to provide effective user experience when interacting with the body directly and reduce ambiguity.

Finally, I present some actual approaches evaluating recognition studies and how prior work used them to ensure the robustness of the systems.

2.1 Capture

There are diverse methods and sensors for capturing information from a scene: color cameras (RGB), infrared cameras (IR or thermal cameras), depth sensors, lasers,

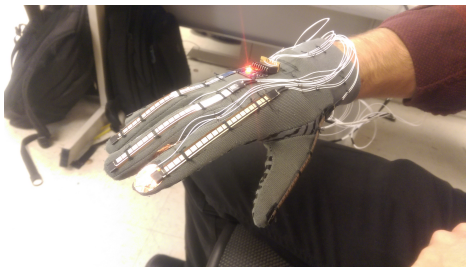
reflective markers, etc. Gathering depends on the sensor type data and is appropriate for some specific tasks and not for others. As an example, in an assembly line which produces water in metal bottles, infrared cameras can detect fluids inside the bottles while color cameras cannot, and therefore when testing if the containers are full, infrared cameras will provide the necessary information.

Since this study is focused on Sign Language (SL), the information of the scene is limited to the hands. To capture hand data a special sensor has been developed to recognize the position of the hand skeleton, Leap Motion [6] controller; compared to other sensors, it is small and portable.It is sometimes attached to a Head Mounted Display to add the user's hand to a virtual world. Because of physical device constraints, the capture area is limited to a modest distance from the sensor; the area of capture restricts the users movement and it may not be appropriate for sign language as stated in [53]. Example applications include: *Shortcuts* [7] is an interface for Human-Computer interaction on Windows and Mac devices, it provides shortcuts for switching applications, volume handling, media player actions, etc; *Cyber Science - Motion* [2] challenges the user to explore, dissect and assemble a skull to learn about human anatomy.

Hand tracking gloves and markers on the hand are other techniques to capture and track finger information for the recognition of hand gestures [19, 73]. Gloves are accurate in determining the exact position of each finger but they are not comfortable for users, they can obstruct fingers' movements and the resulting hand gesture may not be correct. Markers are not uncomfortable but they need a specific setup step and a constructed environment. Because casual users of the system may not

have in possession the required equipment or they may not know the environment arrangement, this approach may not be suitable for the problem.

(a) Leap motion sensor.



(b) Hand glove for hand gesture recognition.

Figure 2.1: Specific sensors for hand gesture recognition.

I focused on capturing images from the scene, the distance from the camera to each point of the hand of the user. The camera captures an image of the scene in which each pixel measures the distance to the closest object. Depth provides features imaging cameras cannot offer, for example, finger positions; color images are 2D images in which each pixel represents the intensity of the light of each color channel at that point, while depth sensors capture spatial distance points.

Falling into the category of depth sensors there are different methods of capturing distance information of a scene. The most common ones are Time-of-Flight sensors[9], Structured light cameras/sensors[8], and Stereoscopic cameras[10].

2.1.1 Time-of-flight (ToF)

ToF cameras (Figure 2.2a) measure the time a light signal spends flying in the space between the camera and the object in front of it. For each pixel of the image, there is a measurement which provides information about the depth of an object. These

(a) Time of flight. (b) Kinect camera. (c) Stereoscopic camera.

Figure 2.2: Sensor types

type of cameras produce accurate depth images at a high frame rate (150 frames per second) empowering real-time recognition systems, but they are sensitive to lighting conditions and reflective surfaces. The resolution the ToF device offers is lower than other sensors [9]. Low resolution cameras may not fit the requirements of the SL problem because it does not properly detect dissimilarities in hand images or the hand may not fit within the boundaries of the image. Using more than one ToF sensor to increase the resolution of the image (stitching images together) increases the resolution but it produces noise in the estimation of the depth as a result of the interferences between them. In spite of the low resolution of the cameras, some work has been done using them for hand recognition [50, 65] obtaining accuracies around 90% in real time.

2.1.2 Structured light

The principle of structured light is based on projecting a specific light pattern into a scene. An infrared (IR) light emitter projects a dense structured matrix of dots (IR

pattern) onto the scene; these projections are captured by an infrared camera. As shown in Figure 2.3, the scene warps the pattern and by triangulation, the position of each dot is calculated. Since the projected pattern is known, the difference between the warped and projected patterns provide information about the scene's depth. Internal processors of the device compare the distortions against the known pattern to estimate the distance value of each pixel and create a 3D point cloud of the scene. These type of sensors do not work well with bright lighting conditions because the overabundant light drowns the projected IR pattern.

Since the light emitter and the camera are different sensors, they are separated by a certain distance which forces the system to set the minimum distance to around 0.6 meters to perform the triangulation. Large distances make the projection sparse and thus the accuracy of the image decreases. These sensors have an optimal capturing range, which depends on the device constraints, but the optimal range of this type of system is approximately 1.2 to 3.5 meters.

Microsoft Kinect, as shown in Figure 2.2b, is a camera which captures the depth images of a scene using the Structured Light principle. Since its release in 2010 for the XBox gaming console, it has been used for many scientific purposes [35, 39, 59, 67]. Regarding hand gesture recognition and Sign Language problems, the Kinect has been used in diverse methods, concluding in high accuracy and real time systems [39, 55, 56, 59].

2.1.3 Stereoscopic

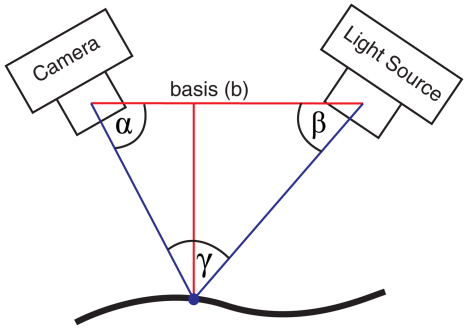


Figure 2.3: Structured light principle, how to measure the depth of the scene by triangulation, using the distortioned infrared pattern.

Stereoscopic cameras (Figure 2.2c) use two cameras with a certain displacement to get depth information. They simulate the biological eye system to measure depth using a 3D triangulation as shown in Figure 2.4. These type of cameras have the sensors embedded in the same device. Since the method of depth estimation is based on the sensor's position and knowing their relative distance, it is not required that the sensors are built-in in the same device. It is viable to get two monoscopic cameras to imitate the stereoscopic ones if the setup of the system is known and appropriate calculations are performed. This method has the same problems as the Structured Light sensors: the field of views of the cameras do not overlap and thus they have a minimum capture distance.

The crucial part of measuring the depth from stereoscopic cameras is to find corresponding points in the two images. Knowing the base length between the two sensors and coordinates of the corresponding points in both images it is possible to compute the depth of it by triangulation. This technique has high computational cost and the resulting depth map has low precision; smooth color objects in the scene do not offer good enough features to correspond to pixels in both images, while

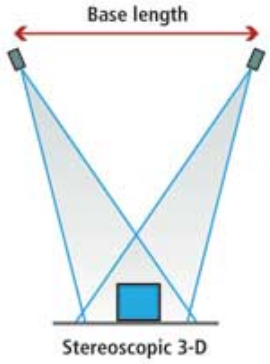


Figure 2.4: 3D triangulation of the scene using two common cameras.

sharp ones do [17]. Therefore, the accuracy of the depth map is determined by the composition, textures, and colors of the scene.

Even though the computational cost is high and the accuracy is subject to the scene arrangement, stereoscopic cameras have positive aspects: they can be built with normal cameras, thus the resolution and the frame rate is variable; and, they work well with different lighting conditions.

Diverse publications showed that using these type of cameras it is plausible to achieve high accuracies for human-computer interaction systems in real time. The author in [68] present a method using Hidden Markov Models (HMM), based on human body images as input to control characters in gaming. Jojic presents in [33] a real-time system for detecting pointing gestures.

I chose as the optimal sensor the Microsoft Kinect camera, as it is inexpensive for the resolution and precision it offers. It has a VGA camera with a 1920x1080 resolution to capture high-resolution color images. It has a QVGA sensor (Structured Light sensor) with a 512x424 resolution: it captures the depth information of the scene. This sensor has an optimal capture range at 1.5-3.5 meters; objects very close

or very far from the sensor are not measured well. The two sensors, VGA and QVGA, work at a frame rate of 30 fps; since the minimum frame rate to get fluid captures is 24 fps it is sufficient to record videos and also work in real time.

I used the QVGA sensor to capture the images. The frame rate is enough for developing real-time systems, as 30 fps is sufficient to track the bodies of the people and segment the hands from them. One advantage of using the Kinect is the built in SDK: the requirement to recognize the human body and track the hands is already developed, so there is no need of carrying out with this work again. Approaches for body tracking [52, 59], hand gesture recognition [35, 39, 47] and Sign Language recognition [14, 35, 39, 47], have taken advantage of this camera and there are a lot of existing documentation and forums. The Kinect camera has capturing problems, but defining a specific environment to record users minimizes their impact.

2.2 Recognition

NNs are popular in image processing over other classification methods because they can, in theory, be trained to perform any regression or discrimination task [25]. These mathematical models are based on reproducing the behavior of biological brains by learning from real-world data instead of hand-coding features: in the case of image processing, the features of the images.

The first attempt to imitate a biological brain was the mathematical model created by Pitts in [49], called *threshold logic*. Based on this idea an algorithm for pattern recognition [57] was created to perform addition and subtraction operations. These basic neural networks were networks of perceptrons (neurons) organized in connected

layers but had limitations in performing some operations until Werbos presented a backpropagation algorithm [70]. The backpropagation allowed the learning of the NN, in adjusting the weights of the layer connections to perform more complex operations. Depending on the problem and the amount of training data, the training step may require high computational cost and time, that is the reason why simpler methods overtook these models in the Machine Learning area.

Nowadays, hardware has more computational power than the early beginning of NNs. Dealing with learning tasks became time-affordable gaining popularity for classification problems. One of the most popular types of NNs for image processing are *Convolutional Neural Networks* (CNN). They are inspired by the organization of the visual cortex in a biological brain: as shown in Figure 2.5, the perceptrons are not fully connected to the next layer's perceptrons, instead, they are connected only to some of them. This idea was presented in a experiment [29] where it was shown how some specific neurons reacted to particular edge orientations. The global idea of a CNN is having an image as an input to the model, passing it through the different layers and getting an output that can be a single class or a probability of classes. There are three types of operations in a CNN: the convolution step computes a convolution over the input image using the already learned feature maps as filters; the pooling step reduces the dimensionality of the convolved image; and the fully connected layer works as a common neural network, where each neuron is fully connected to the neurons in the next layer.

The CNN and its variations have been used across different areas in diverse problems[13, 24, 41]. One of the areas is Natural Language Processing (NLP). In

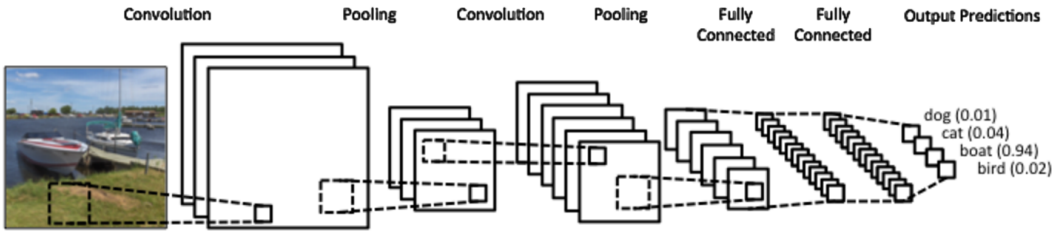


Figure 2.5: The structure of a CNN.

[22] a single CNN architecture is presented to provide information about a sentence: part-of-speech tags, chunks, semantic roles, similar words, etc. Using a CNN, Cicero [24] performed sentiment analysis of short texts using Twitter posts. Text classification is performed in [40] using a Recurrent CNN, capturing contextual information of the text. Another use of this type of neural networks is in the area of speech recognition, due to the spatiotemporal feature of the CNN they are suitable for these kind of tasks. In [13], a hybrid system of a neural network, a HMM, and a CNN is presented for multi-speaker speech recognition. Continuing this research line the same author shows in [12] his experiments on how to use a CNN for speech recognition and compares the result against a standard hybrid Deep Neural Network-Hidden Markov Model on the TIMIT recognition tasks.

Among the various areas a CNN can be used in, the most popular one is image processing. In this case, the CNN learns from a training image set and creates feature maps which, after a convolution, detects local features. These features are kept for the next layers of the process to recognize more abstract features by combining simpler ones. This process reduces the dimensionality of the image and thus the efficiency of

the neural network is increased. In [41] diverse methods for handwriting recognition are presented and the use of a CNN is studied; these studies show how the CNN eliminates the need for hand-crafted features. The results reported in [24] demonstrates the robustness of a system for detecting handwritten letters on bank checks, integrating it across several banks in the US. Some work in breast cancer detection using mammography images has been done in [26, 45, 58]. Magnetic Resonance Imaging (MRI) images have been processed with a CNN in [15, 54, 75] to detect the gray mass in the brain and identify neurological diseases. Challenges in image recognition [5] taking advantage of NNs was raised due to the actual computational power; given a dataset with millions of images challenge participant teams had to recognize objects in the images. The datasets of the challenges are public and scientist used CNNs to create recognition systems [3, 5]. There was an inflection point in 2012 when the CNN presented in the work [38] won the challenge ILSVRC2012 [5] reducing the error rate from 26.2% to 15.3%. With the power of two GPUs, they designed and trained a CNN called *AlexNet*[38] to deal with the 15 million labeled images.

Other recognition models have been used for image processing and automatic human body detection which are the main features of diverse applications: virtual reality, surveillance, person identification, fall detection for elderly people, or gender classification[21, 43, 60]. These applications segment human bodies from environments with varying conditions such as lighting, weather or a number of objects. Once the system is capable of detecting bodies the next step is tracking them to capture the behavior of people. As an example, video surveillance with human body tracking can be found in London[21], where the use of RGB cameras in a network all over the

city permits the tracking of people.

Another use of body segmentation is body joint estimation to recognize the skeleton and body pose. The detection of different parts of the body empowers Natural User Interfaces for Human-Computer Interaction (HCI) applications. Since the release of the Microsoft Kinect camera, the use of bodies for controlling characters in video games has risen[28].

2.2.1 Detection and extraction from the environment

The detection of human bodies in single images or a sequence of frames is still a challenge for scientists and engineers because of the changing conditions in the environment[62, 75, 77]. Depending on the camera types used to capture images or videos, the weather has a great impact. Sunny days ensure colorful images while cloudy or rainy days decrease the brightness and the contrast of them. Such changing images require robust surveillance systems to deal with diverse lighting conditions. People wear different clothes with a variety of colors, textures, and shapes, and perform different poses while walking, increasing the difficulty of human body detection [77].

A study of low lighting conditions in videos for body detection and segmentation is presented in [62]. The authors proposed a method to extract the upper human bodies from the background. First, the system learns how the background looks. When an input image is fed to the system, it computes the subtraction (difference) from the background giving as a result a shape. Features are extracted from the shape to feed a Support Vector Machine (SVM) to detect whether or not the shape belongs

to a human body or not. The second procedure relys on detecting the contour of the shape assigning an energy function to the region and minimizing the error.

Some authors used thermal cameras to detect and segment human bodies from the images[31]. Since the human body keeps its internal temperature in a small interval the detection it is done by thresholding. Weather conditions make the detection trivial because the surrounding areas of the person are colder than the body[31], but on hot days the difference between bodies and background is not as simple. Moreover, people use clothes which block the capturing of heat of bodies [77]. In [48], a rescue robot system is presented which is capable of switching between teleoperation and autonomous navigation. The robot is equipped with various sensors including a thermal sensor. When the robot cannot communicate with the central unit to be teleoperated the autonomous system takes the lead guiding the machine to the location of the person at risk.

Robots capable of navigating through a crowded environment require the detection of humans around them to avoid collisions. This problem is studied in [74] where authors used color-depth cameras on a robot to detect multiple human bodies in real time (as stated in [66] 4 fps are enough for real-time decision making). The ceiling and the floor do not add crucial information to the scene, the depth point cloud is preprocessed to eliminate them using an extension of the RANSAC algorithm. The authors introduced a new concept for multiple human detection called Depth Of Interest (DOI) which is analogous of Region Of Interest based on the depth, where each DOI is a highly probable interval of human or objects in the depth map. The probable regions are fed into a decision Direct Acyclic Graph (DAG) based algorithm

to handle the detected candidates and track them using an extended Kalman filter[74].

2.2.2 Body tracking

In [42] a video surveillance system is presented to detect human abnormal activities. Object tracks are received in real time and those trajectories are fed into an event analysis module which determines whether or not an outline activity is being observed. The surveillance system requires the user to input the illegal areas for pedestrians. An intensive video analysis algorithm is applied to decide if the alert is being triggered by a human being or an object. A network of cameras for human body tracking is presented in [20], a set of cameras are dispersed around a city without calibrating them. The aim of this work is to calibrate all the cameras in such a manner that they are capable of covering neighboring streets: when a new camera is added to the network a recalibration is performed to extend the covering area. This way, while a person is being tracked by a single camera and it exits the field of view of it, another camera will detect it as the same person.

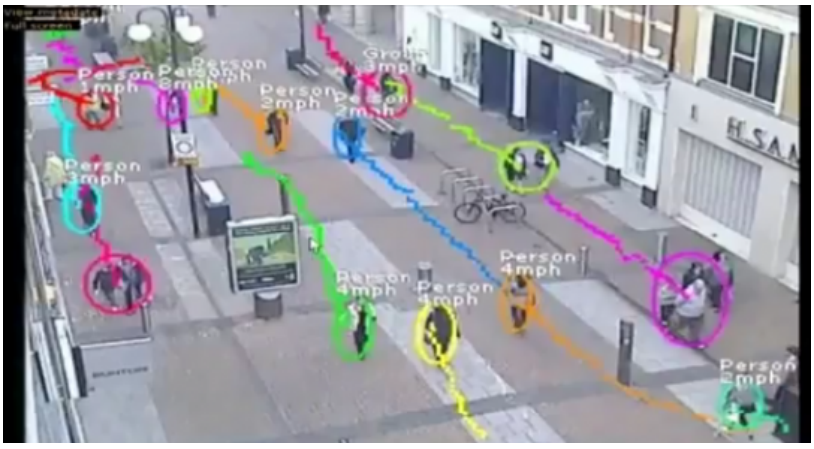


Figure 2.6: A surveillance system in a street.

Using a RGB camera mounted in a drone Imamura presented a human body tracking system in [30] to create a drone which follows a person. The author avoided using color differences and used Histogram of Oriented Gradients (HOG) [23] features to feed a linear SVM. As shown in Figure 2.2a, HOG divides the image into small cells to compute local histograms; blocks of adjacent cells are compared to get the direction of the gradient. Once the human body is being tracked by the drone it follows the person at a certain distance.

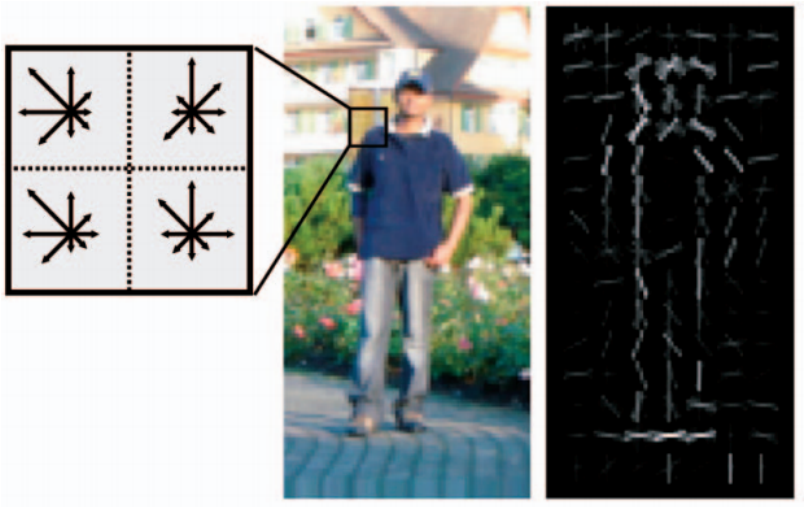


Figure 2.7: Histogram of Oriented Gradients of an image containing a person.

2.2.3 Body joint extraction and tracking

One of the next steps of body recognition once the body has been segmented and tracked from the original image is to estimate body joint positions. Joints provide information of spatial location of body parts; knowing the joints' relative positions it is possible to determine the body gesture.

In [59] a method to extract body joints using Random Decision Forest (RDF) from the Kinect camera's depth images is presented. Taking advantage of the depth map, random pixels are selected as features of the image. Then a comparison is performed by randomly selecting close pixels to compute the difference; this technique provides features about spatial arrangement of pixels. To determine in which part of the body the pixel lies on, the random selection is done several times with a large set of parameters. The configuration which provides the maximum information is selected as the splitting approach of the node in the RDF. As stated in the study, these individual features are weak to classify each pixel but using all of them the classification is accurate. This procedure segments different parts of the body. Computing the barycenter of these parts provides the joints to track the human skeleton as shown in Figure 2.8. Another approach for full body pose reconstruction using a depth camera is presented in [16]. The presented variation of Dijkstras algorithm provides features used for matching the 3D point cloud from the ToF camera to a synthetic pose dataset.

Figure 2.8: Body joint estimation procedure.

2.3 Hand gesture recognition

Hands are used more often for communication between humans than other parts of the body. While two humans are talking there is a non-verbal communication using body gestures and hand gestures to express the meaning of the speech. We unconsciously use our hands for interaction with other humans, the use of them as a more natural form of interaction between a user and a computer is studied by Karam in [34]. Nowadays, interaction of human users with computers has increased due to the regain popularity of Virtual Reality devices. Diverse sensors and approaches have been developed to capture hands.

The first step for hand gesture recognition is to segment the hand and detect the main features of it. A simple technique is presented in [47] where the hand's joints are received from a 3D camera and applying the K-Nearest-Neighbor algorithm concludes in a recognition rate of 97%. Using the same technique as [59] for body joint estimation, Keskin extracted the hand skeleton in [35]. In this case, a synthetic dataset was created from 3D hand models to train the RDFs; the hand models enhance the possibilities of generating additional with more variation. For testing purposes real data was collected, the real dataset was fed as input to the RDF (trained with synthetic data), achieving accuracies above 99%. Another method is proposed in [39] where a method to segment the hand from the body using Connected Component Labelling is presented. The scene is labeled based on connected pixels. Pixels lying in the same depth range are assigned with the same label. The closest group of pixels are selected as potential candidates for the hand; it is expected that the closest object to the camera are the hands. The hand values are normalized, scaled and rotated to

fit into a normalized hand image because the same problem as in Section 2.2 for face recognition exists.

Vo presents in [67] a dataset of depth images for Vietnamese Sign Language. The images enclose only the hands with no background. Therefore this dataset is useful for my purposes and I used it to perform my study and evaluation. The authors presented a method to recognize images based on the standard deviation and the averages. The images are divided into a fixed number of square cells, and for each cell, the standard deviation and the average is computed. Cells are sorted in a descending order and the resulting feature vector is fed to a SVM. The results achieved between 94% and 100% of correct recognized labels.

Systems to interact with computers in a natural way using only the users hands have been developed. In [55] features to segment the hand are computed using Gabor filters: the combination of a bank of Gabor filters empowers features of hands in RGB-D space for recognition purposes. The system is trained using American Sign Language letters as the training set for multiclass RDFs. The authors designed a GUI to interact with the system: while the user is performing a symbol, the system recognizes a set of possible letters and displays them in a semicircular arrangement, as shown in Figure 2.9. In [51] an interface for human-computer interaction is presented. The system uses an RGB camera together with depth information to track the hand and recognize the gestures.

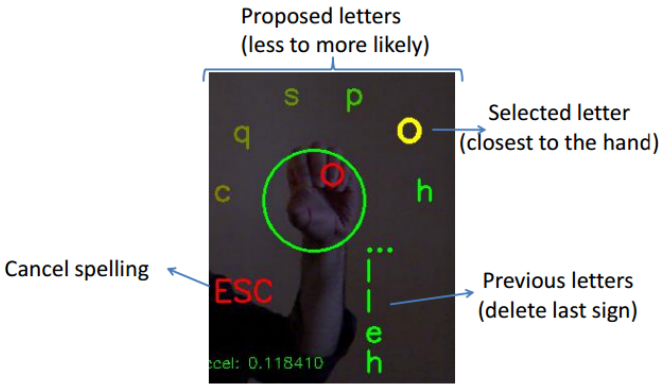


Figure 2.9: GUI from [55].

2.4 Face recognition

Together with hand gestures, facial expressions are used. Facial expression is important in SL, the meaning of certain hand gestures is conditioned by it. Face recognition has been studied in different areas and trough diverse methods to segment the face and recognize its expression. Surveillance security, border control or digital entertainment are examples of possible applications [21, 61]. Among these applications, the main problem is the pose of the subject to be recognized, since the person's focus is not the camera and the direction he/she is looking at is arbitrary; there may be self-occlusions or high variances in the intensity of the light. The center of attention of the scientist is on solving these problems, thus they have performed different studies and presented diverse approaches to deal with them.

Early algorithms for face recognition [63, 64] assumed the correspondence between face images, and therefore the relative position of eyes, nose, mouth had little variance. The method used in these algorithms relies on computing the eigenfaces by performing

Principal Component Analysis (PCA) of data and creating an average face. Results showed high accuracy for frontal face images, but for images with arbitrary facing directions (as shown in Figure 2.10) the precision decreased. Algorithms solving aleatory facing directions were presented in [18] which take into account spatial correspondence of facial features. Works have been proposed to avoid using facial feature correspondence [69, 76]. In [69], the approach was to detect key points of faces, using SIFT feature detector, and matching them to faces in a database. This technique allows the partial detection of faces when they are occluded by objects in front of the person. In [76], deep learning techniques were used to detect faces and reconstruct them in face frontal images, this is, if the input face is not facing the camera or there are occlusions, the system can transform it to reconstruct the missing parts and put it facing the camera.

Figure 2.10: Different facing directions in face images.

As soon as the face is located and features have been detected the process for face expression recognition starts. As stated in [36], facial expressions are deformations of face features (eyebrows, mouth, etc) so the extraction of these features is a must. Displacements of face features or changes of face texture are the most used characteristic to determine the final feature representation of it. Diverse machine learning algorithms can be applied to the resulting feature vector: K-Nearest Neighbor, Artificial

Neural Networks or SVM [46, 72, 44].

2.5 Evaluation

The system LeCun [41] trained on the MNIST dataset with 60000 examples of handwritten digits, achieved 82% of recognition rate on a test set of 646 machine printed business checks. Compared to previous systems, with a recognition rate of 68%, tested on the same dataset, the improvement is considerable. Compared to previous work, in [38] the *Alexnet* NN was trained over millions of color images from the *ILSVRC-2010* [4] challenge dataset. The authors tested the system using the top-1 and top-5 recognitions, achieving 62.5% and 83% respectively.

In [62], the dataset used for training the system to segment upper human bodies from low contrast images was a dataset created from ATM video records. To evaluate the system the authors used precision and recall metrics, obtaining an average of 98% of precision and 96% of recall. In night videos the recall dropped considerably due to low contrast.

From depth images captured with the Kinect, Shotton [59] evaluated the body joint detection system using a synthetic test set of 5000 images and a real test set of 8800 images. Results showed a mean average precision of 73%. The pose recognition system presented in [35] used 4000 frames from videos captured by a ToF camera to test the approach. 86% of the images were recognized correctly. Using the same technique as in [59], Keskin trained a system in American Sign Language digit dataset with real and synthetic images. The system obtained 99% accuracy on the synthetic dataset with 20000 images. On the real dataset, a collected dataset with 300 frames

per gesture, the authors achieved 98% accuracy. In [39], the authors captured their own dataset of twelve gestures from American Sign Language to evaluate the system. Each gesture was performed 3 times by each of the 10 participants. The results shown an average recognition rate of 83% for the diverse models. The Vietnamese Sign Language dataset presented in [67] was used to train and evaluate the authors approach. They used 5 models with diverse parameters to test the method, achieving an average precision of 98%. In [55], the gabor filter approach was evaluated using a self collected dataset of 48000 images. Half of the dataset was used for training an the other half for testing, obtaining a mean precision of 75%.

CHAPTER 3

METHODS

Using image processing and image classification techniques it is possible to recognize hand gestures in images. The process of hand gesture recognition is divided into hand location, hand segmentation and its classification. My method locates and extracts the hand to generate new images, uses them in a Convolutional Neural Network (CNN) to classify and then recognize hand gestures. To build a classification model it is required to get training data, by segmenting a user's hand, from the Kinect. Using previous knowledge and tools, I conducted a preliminary study to detect the constraints of the problem. Results provided information about how to select better training data and a more suitable classification model. I used this information to improve the system's recognition. To determine if a dataset has enough variance to generalize the problem, I developed a method based on [64]. I used CUDA for parallelization to increase the performance of the method.

The complexity of the scene (colors, reflective materials, and other objects) takes an important role in the capture. The capture of images may not be good enough to detect, recognize or classify signs. I created an "ideal room with no elements that affects the image capture using the Kinect. When capturing each image I used the camera's depth data. The user's position was crucial because the depth sensor had a limited optimal range of capture. This range was between 1.5m and 3.5m. I set the user seated at a distance of 2m in the center of the field of view of the camera. I used the captured images to extract only the hands for processing using a classification model.

Image classification is the task of taking an input image, processing it and assigning a unique label (or a class) from a set of labels. The result may be a label

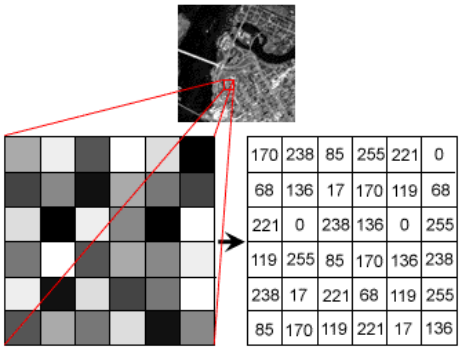


Figure 3.1: How the computer "sees" an image.

or a probability of belonging to diverse classes. Humans, are able to easily recognize objects in images. We can adapt to different environments, recognize patterns and generalize from our knowledge for determining what is it in the image, but computers do not have these skills.

When a computer takes an image as an input it sees a matrix of pixel values, as shown in Figure 3.1. Depending on the number of image channels (RGB = 3 or Grayscale = 1) pixels are unique values of channel length arrays, that describe the intensity of the pixels at each point. If the image is Grayscale, pixels have an integer value in the range [0,255]. For RGB pixels having an array of length 3, one for each color with same value range as Grayscale [0,255]. The idea of image classification using machines is to use these values to detect patterns in the image and provide a resulting label or a probability of the input belonging to a class.

3.1 Hand identification and segmentation

In Section 2.1 I explained the advantages of the Kinect camera and how developers released the Microsoft Kinect SDK (KSDK) to work with the sensor. This tool is capable of tracking a user's body joints. I used it to get the location of the hand. The KSDK provides four different joints in the hand for its tracking: wrist, palm, thumb and finger tip. I only used the palm joint for locating the hand because it was the joint in the hand with the least location error.

The Kinect camera has sensors with different resolutions for capturing different types of images. The coordinate spaces of each individual sensor's data is different and one element from a sensor cannot be matched directly with the same element from another sensor because they are not aligned. The KSDK contains functions to perform alignment of data in different coordinate spaces. I aligned the joint coordinate system to the depth sensor's coordinate system in order to produce images to locate and segment the hand.

To segment the hand, I had to choose the size of captured images. I conducted experiments to see what was an effective size: since I have defined an ideal world, the users do not a vary in position, and thus, the hand is of the same size (only with small variations). The effective size for the problem was 100x100 images: these images enclose the hand and have room for small movements.

A possible issue arise when the user's hand is close to the borders of the camera's field of view, the Kinect cannot track the entire hand. Because I limited the capture window to the sensor's field of view this can provide images in which hands are partially segmented. My method, based on local features, recognizes some partial

hands using the characteristics of gestures but not all.

The extracted window from the image provides all the data in that region, which means the background is captured too. I used depth data to differentiate background and foreground objects. As stated in [39] when capturing users performing Sign Language, the closest objects to the camera are the hands. I used the same thresholding technique as the authors in [39]. In this case, I know the location, and thus the depth, of the hand. I defined two thresholds [39]: one for objects in front of the hand and another one to discard the background. Pixels with the values that do not lay inside the rectangular cube defined by the image size and the two thresholds are assigned the value zero, as shown in Figure 3.2.

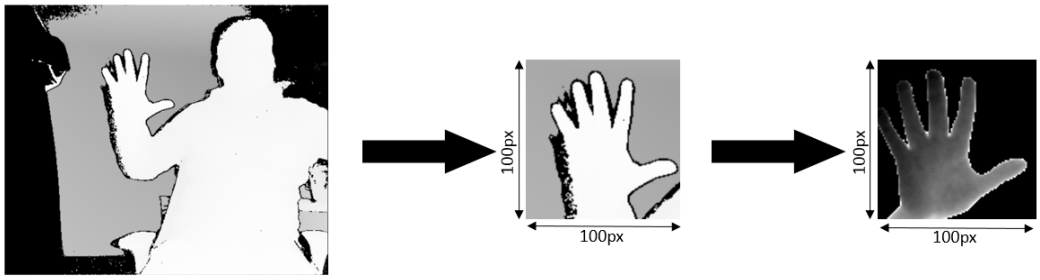


Figure 3.2: Hand extraction process.

3.2 Preliminary experiments

My first hypothesis was to use Nerual Networks (NN) and Random Decision Forests (RDF) for recognition of raw images of number signs. To evaluate the models, I used a dataset of my own hands. The dataset contains frames of a video sequence of me

performing hand symbols for numbers 0-5. The hand, as I explained in Section 3.1, was cropped from the camera in images of 100x100 pixels and thresholded to segment the hand. The values of the images were the raw distances from the camera to the hand, no normalization methods were applied to them. To add some variation to the gestures I rotated the hand and I moved it forward and backward. The dataset has a balanced number of instances for each class (0 to 5), around 700 images per class.

In the next sections, I explain the two classification models, RDF and NN, I used for testing my hypothesis using the collected hand dataset. In both cases, I used WEKA [71], an open-source Machine Learning software. It is based on Java and it cannot use input data directly from the Kinect so I used it to perform the first tests. Once the models were trained I used them for recognition of the frames of the Kinect. I created a bridge between JAVA and C++ using the Java Native Interface. This communication is too slow to work in real-time but it helped me to gather information about the behavior of the models with real data.

3.2.1 Random Decision Forest

A decision tree is a model which takes into account the importance of each variable in the dataset to split the decision path in two [59]. The variable providing more information is selected as a threshold at the first node to perform the division of the data. After each split, more nodes are added to the model using the same approach to create a tree like structure. When input data is processed, the program traverses the tree making a decision in each node based on input features until it reaches a leaf node. As shown in Figure 3.3, to traverse the tree, at each node it chooses the left

or right path depending on the variable and threshold values. Once a leaf is reached, there are no more splitting nodes and thus it labels the input with the predicted class.

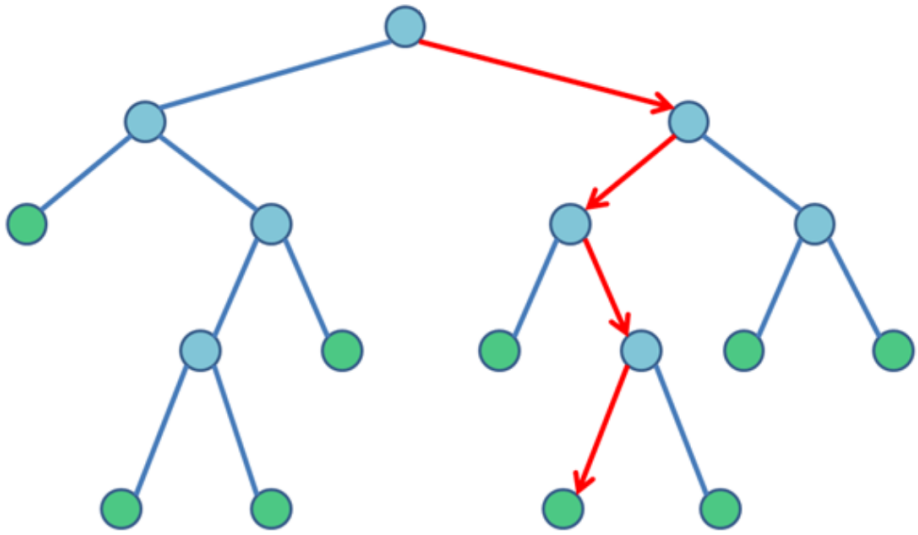


Figure 3.3: Decision tree model. Blue circles are nodes, green circles are leaf nodes and the path in red is the decision path to reach a leaf and determine the label of the input.

As shown in 3.4, grouping a set of decision trees trained with the same data but with different strategies provides a resulting array of labels which avoids the tendency of the decision trees of overfitting. The most common strategy to train them is the random selection of a set of features and select the one which provides more information, as it is done in [59]. For classification purposes, the mode of the label array is selected as a final label.

For this first study, I created a RDF containing different number of trees with a maximum depth of 20 layers. Initially, it was anticipated that large numbers of trees, in the order of 1000, would be necessary for generalization of the problem

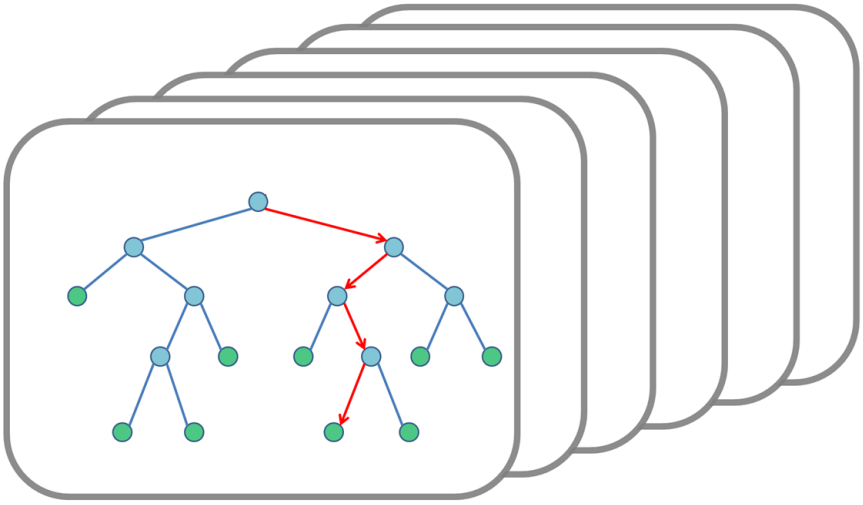


Figure 3.4: Random Decision Forest.

and thus a recognition rate around 80%. However, despite constraints and size of images, I got high accuracy (99%) performing a 10 cross-fold validation [37] which lead me to reduce the number of trees. I set the forest with 10 and 100 trees and still achieved precisions, with 10 trees I achieved 85% accuracy, this result was promising. These good results generated some ideas (explained in Section 3.2.3) about what was happening, as such good results were unexpected, because the number of pixels in the image and the variations are too high to generate such a high accuracy with shallow trees. The models may be overfitted.

3.2.2 Neural Networks

Neural Networks, as I explained in Section 2.2 are based on connected perceptrons organized in different layers. The neurons in the same layer are not connected between them, they are connected to the next layer. The most common and simplest ones have three layers, as shown in Figure 3.5: the input layer (this layer has one neuron per each variable in the feature vector), the hidden layer (depending on the problem, the number of perceptrons vary), and the output layer (it contains one neuron per class in the dataset) which determines the class of the input data.

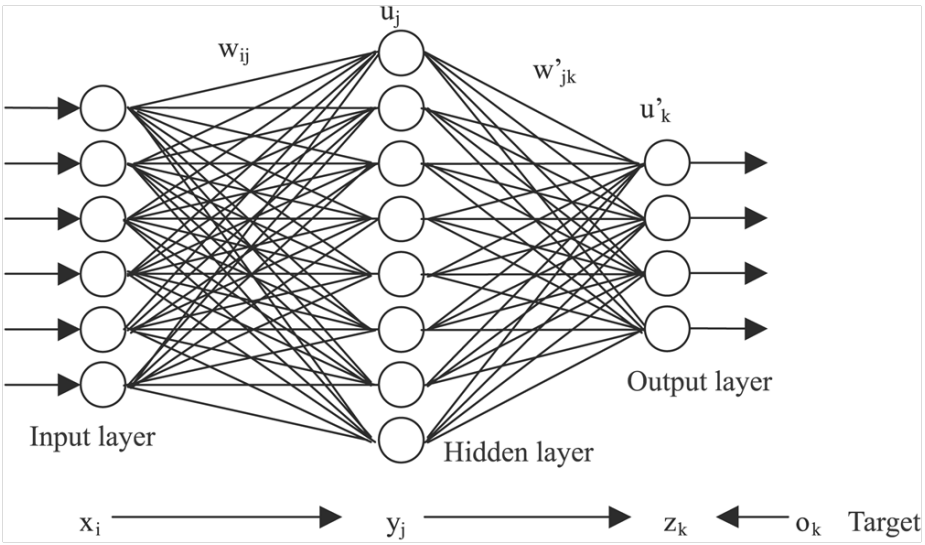


Figure 3.5: A three layer neural network.

For this work, I chose hidden layers with a few number of perceptrons to train them fast, in order to see which kind of results I could get and see the system working even though the precision was low. In the first test I randomly set the number of neurons in the hidden layer to 20. I used the pixels as features of images, so based on

the size of them, the input layer size is $100 \times 100 = 10000$ nodes; the hidden layer has 20 neurons; and the output layer the number of classes, this is, 6 nodes. The results after a 10 cross-fold validation were surprisingly high: I got 85% correct classified instances.

I tested different numbers of hidden neurons to analyze the behavior of the network. The accuracy decreased but not as much as I thought. When using 10 hidden neurons I got 67% accuracy and while using 5 the resulting precision was around 32%. These results were good for the number of neurons used in the hidden layer. Compared to the size of the problem, the dimensionality of the feature vector is big for the number of hidden nodes.

I increased the number of perceptrons in the hidden layer up to 100 and the results showed high accuracy, it reached 99% of correct classified instances. Again, using a small hidden layer and achieving high accuracy, as happened with RDFs in the previous section, provided information about something wrong happening.

When I used the NNs for recognition in a real environment, the system could not recognize correctly any gesture performed by the user. This fact together with the results of the RDFs confirmed my hypothesis of something wrong in the system and led me to the conclusion that the dataset used to train the models was not suitable to train them.

3.2.3 Preliminary results and learned lessons

After evaluating the two models using the hand number dataset and a 10 cross-fold validation, the results were promising: with NNs received accuracies about 85% while

with RDFs got around 95%. These results showed that the two models should be suitable for this recognition task. I used the NN to recognize the gestures using real-time data, and showed poor results. It was very hard for the system to determine which gestures the user was performing and often the label assignment was misclassified. The same problem happens with the RDF, so at this moment these models were not appropriate to recognize the gesture numbers of the users. This means the models learned key features of the dataset but they were overfitted to the training data, making it hardly possible to recognize new instances outside the dataset.

With such poor results, I concluded three things: (1) images were not normalized and had dissimilar pixel values. When the models tried to recognize the input, they were different from images used for training. (2) since images were frames of a video, even though I did some movements to variate position and orientation of the gesture, the images looked similar, therefore a dataset gotten without taking videos is required (as it is explained in Section 4.1). (3) I had to use models which focus on local features instead of the entire image.

3.3 Approach

Based on the results and conclusions in the previous section I developed two new methods. The first is similar to the approach used in [64], it is based on creating a representative image for each class in the dataset and comparing the input image against them. To increase the efficiency of the method, I took advantage of the GPU for parallelization using CUDA. For the second method, I used a NN which focuses on local features, a CNN.

Regarding the dataset, I searched for already published datasets for Sign Language (SL) and I found a Vietnamese hand letter gestures dataset. I performed tests on it until I realized that the images were frames of a video. Therefore this dataset was not good to evaluate my system, so I created my own database for American Sign Language.

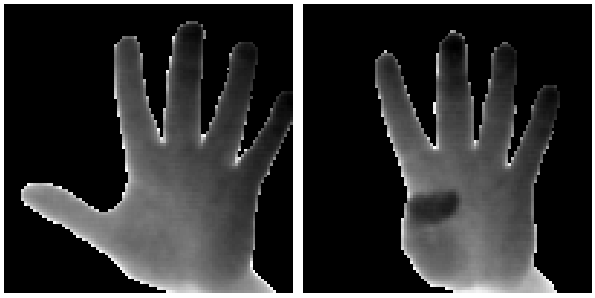
3.3.1 Operations between images

Images provided by Microsoft Kinect camera are images in which each pixel value is an estimation of the distance from the camera to the closest object. These measurements provide information about topological arrangement of the captured scene; in this case, depth information of a given user's hand. Depending on a user's gesture, the depth map has characteristics which represent the gesture in spatial domain.

As an example, in Figure 3.6a, the hand can be seen entirely by the Kinect camera; all fingers are extended and thus the symbol for number five can be recognized. If the user closes the thumb over the palm, as shown in Figure 3.6b, pixels for the thumb in Figure 3.6a turn 0 and the area of the palm is disturbed. These variances represent features of each gesture and the ones the system learns to recognize each symbol.

Since the depth map provides information about 3D space arrangement of the scene it can be interpreted as a 3D model or as a discontinuous and discrete 2D function; as stated in [27] images are discrete 2D functions and depth maps can be considered as a greyscale image.

One of the approaches I studied was raw comparison of images based on [64]. Taking data of image pixels I performed image operations to measure difference between



(a) Symbol for number five. (b) Symbol for number four.

Figure 3.6: Depth map for number gestures.

the input image and the images in the training dataset. I computed the difference between two functions and the resulting function provides required information to know how similar the images at each pixel are.

Images are discrete functions [27] so the sum of all values from the resulting difference image gives the total error of the input image against an image in the dataset. Dividing the error by the number of pixels in the image gives the average error. I performed this operation against all dataset images and the result was an array of error values for each image. Selecting the minimum value from the array procures the most similar image.

Different metrics from the resulting error image can be used to provide a unique metric that gives similarity of two images: *average error*, *min square error*, *standard deviation*... In this case, I selected the *average error* because it has the lowest computational cost from the presented metrics and it can smooth sensor errors or outlier pixels.

This approach is not efficient because the number of comparisons against the

dataset is high and it is not useful for being conducted in real time. In order to achieve real-time performance it is necessary to reduce the computational cost. As shown in Figure 3.7, precomputing an average image for each class in the database provides a representative image of the gesture; it decreases noise in images while the representative features of gestures are kept. Instead of comparing the input image to each image in the dataset I only compared one image per class. This is an improvement in computation time and in memory storage because I only store and compare one image per class.

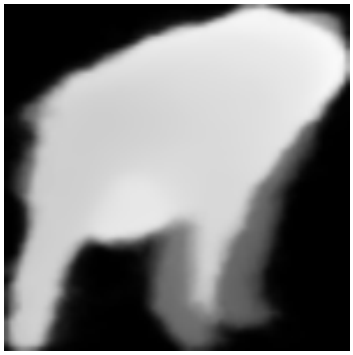


Figure 3.7: Average hand.

CUDA + Reduction sum

To improve the performance of operations between images I used CUDA, which allows to use the GPU (only Nvidia devices) for parallelization of the method. In this case, image operations are pixelwise independent calculations, this is, calculation of the difference of two pixels do not affect neighbor pixels. Depending on device constraints (CUDA cores) and size of the image, in this case, a 100x100 pixel image as it is defined

in Section 4.1, computation time of operations is done in constant time. I used the resulting image as a one-dimensional array so I can use a reduction sum on it to get the total difference error.

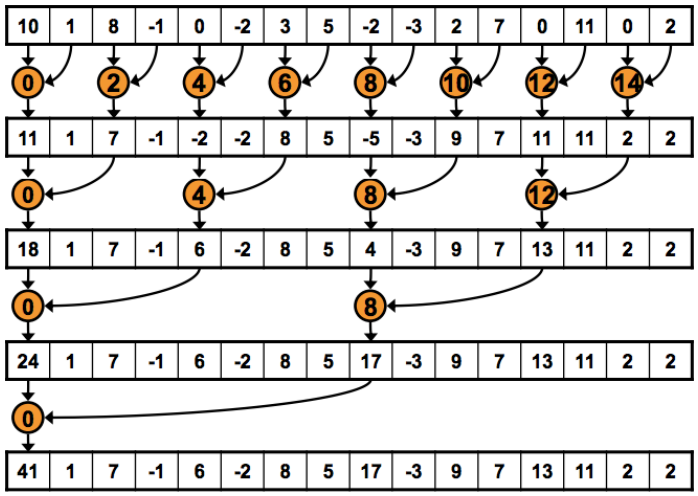


Figure 3.8: Reduction sum of an array using multiple threads.

As shown in Figure 3.8, processing time to add all the error values in an image is $O(log_2 n)$, where n is the number of pixels in the image. In each step, a pair of values is accumulated and stored in the array positions multiples of 2^i, where i represents step count. The maximum steps needed to compute the sum of all elements is $log_2 n$ and the final result is stored in the first position of the array.

Working with CUDA for this problem has some disadvantages depending on the number of classes of the problem. The GPU has its own memory for processing input data, therefore, image data has to be copied from CPU(host) to GPU(device), and this step takes time. Once the image has been processed it has to be copied back from GPU to CPU. Consequently, the total amount of time required to compare

two images in the GPU is $n \times (memoryCopyHostToDevice + processingTime + memoryCopyDeviceToHost)$, where n is the number of representative images.

Time taken using this method is less efficient than doing it using a single thread in the CPU, due to time spent copying data from memories in each device. A solution to that problem is to copy, in advance, all representative images to the device's memory. Now, the only image that has to be copied is the input image; since all representative images are loaded in GPU memory, comparison takes only the processing time. The total comparison time is $memoryCopyHostToDevice + n \times processingTime + memoryCopyDeviceToHost$, which shows higher performance.

It is unlikely to have problems with this last solution but there exists the possibility of running out of memory in GPU. These devices have limited memory and thus loading lot of images may reach the limit. I used 100x100 pixel images with 8-bit depth, so the maximum memory size of an image was 9kB. Actual GPUs have 1GB, 2GB or even 4GB of memory, therefore the number of 9kB images they can load at the same time is enough to handle the problem. Images can be bigger than the ones I used; in resolution, in bit depth and channel amount. In the extreme case that the number of gestures in the database is big enough to fill the entire memory of the GPU, the same case as before arises: it takes more time to copy all the data from the host to the device than processing it in CPU.

3.3.2 Convolutional Neural Networks

A convolutional layer is the first layer the input data faces when it enters into the CNN. The task of convolution in an image needs three elements: an image to convolve, a convolution filter and a convolution stride parameter. As an example, an image of 24x24 pixels as input; the convolution filter is a smaller image learned from training the network, let's say it is a 5x5 matrix. Starting from a corner of the image, the filter is mapped to a region of the image called a receptive field, as shown in Figure 3.9. Each pixel in the image is considered a neuron and the values of them are multiplied by convolution filter neurons; the resulting values are added and a single value is computed to store it in the feature map. After convolution is performed, the receptive field is shifted based on the stride parameter to cover the entire image. Multiple convolution filters lead to multiple feature maps, increasing the recognized features. The training of the CNN focuses on learning the value of neurons of convolution filters, called weights. At the very beginning, these values are randomized and after applying backpropagation algorithms, the network updates the weights to fit training data.

After the convolution is performed an activation layer is used to activate neurons using different strategies. A usual activation layer uses a sigmoid function to activate the perceptrons but since 2015 *Rectified Linear Unit* is used in deep neural networks because it speeds the training stage over other common activation functions [38].

Once feature maps are created from the convolution layer, the next step is the pooling layer. This layer reduces dimensionality of the image while strongly activated pixels are kept. The input to pooling layers are the activated feature maps and

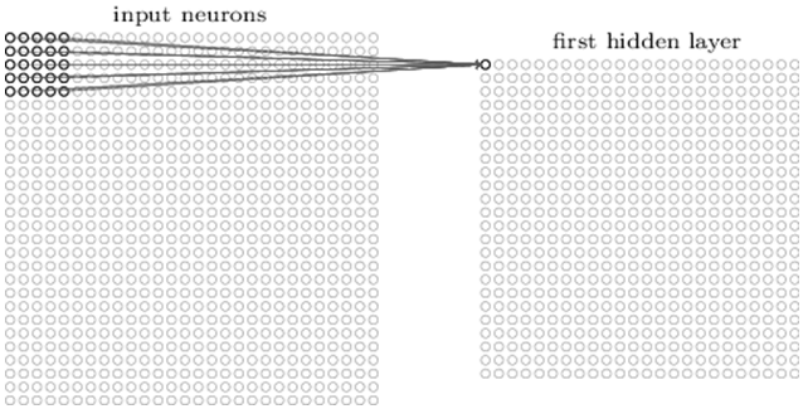


Figure 3.9: Receptive field mapping.

the pooling window size, usually a 2x2 window with a stride of 2. Pooling window computes the max (or average, depending on the used strategy) of each activation layer and maps into a single neuron, then the window is shifted using the stride parameter. This means that the feature maps are shrunk and activated features, which are characteristics of images, are kept.

After processing the input through different combinations of the convolution and pooling layers, the feature maps are images of size 1x1: a single value. These last activation maps are inputs for the last layer of the CNN, a fully connected layer. This layer works as a common NN to produce the predictions of input image, and thus the number of output neurons is equal to the number of classes.

CNN libraries

Tiny-dnn [11] is a library which offers a C++11 implementation of a N for deep learning. It does not require the use of GPU for processing and, as the authors

stated, it is reasonably fast: the constructed sample CNN gets 98.8% of accuracy on MNIST dataset [41] after training for 13 minutes in a Corei7-3520M processor. One of the best points of this library is that it has no dependencies and it is portable, as long as the system where the developer wants to use it has a compiler which supports C++11. The only requirement is to include header files the library provides. It has all the necessary tools to create diverse structures of NNs like fully connected networks or CNNs: in the case of a CNN, it has locally connected layers to perform convolutions and the pooling layers. This library supports models created using the Caffe library [32].

$$input_{CNN} = p_i \times \frac{max_{CNN} - min_{CNN}}{max_I - min_I} + min_{CNN} \tag{3.1}$$

The library I used for recognition requires input values in the range [-1,1] to process data and return an output array. Images in the dataset had 8-bit depth which means each image had 256 intensity values per pixel. I converted the intensity range [0,255] to the required input range of the CNN. To map the intensity range into the required range I used Equation 3.1, where p_i is the value of pixel i in the original image, max_{CNN} and min_{CNN} are the max and min values of the CNN range input, and max_I and min_I are the max and min of the intensity range.

Caffe [32] is a framework for deep learning developed by Berkeley Vision and Learning Center (BVLC). Using an expressive architecture the construction of neural networks is relatively easy; it does not require hard-coding to configure the network. One of the best points of this framework is the use of GPU (CUDA and cuDNN [1] - a CUDA library for deep neural networks) for creating models, parallelization

of training increases the speed to produce the final model. As the authors state, Caffe can process 60 million images per day using a Nvidia K40 GPU, it was used in ILSVRC2012 [38] challenge, creating the winning model for SuperVision section.

I used Caffe framework for training the network and I used Tiny-dnn for loading the model and recognizing input images from a video sequence.

Neural network design/architecture

As I explained in Section 2.2, CNNs learn different local key features from images while they reduce the size of them in each step. Detected key features are kept for the next layers until they reach the fully connected layer. At this point, feature maps are composed of single elements (1x1 images) in which the value of each one provides information about a learned feature of the image. Combining these values a feature vector is constructed to feed the fully connected layer for classification purposes.

The input hand images in the datasets (Vietnamese SL and American SL letters) I used have a size of 100x100 pixels in which each pixel is a distance measurement from the Kinect to the closest object in the scene. At the very beginning, I used the Tiny-dnn library due to the CNN creation method. It showed good results using the Vietnamese SL dataset defined in Section 4.3 when I used the American SL dataset, training time was high and accuracy low: 3% (random classification). I changed the library to Caffe which was more efficient at training stage and got higher accuracies.

The strategy I used to design the CNN architecture was to keep the images and feature maps in appropriate sizes to convolve and pool to reach the last layer. Adding more layers raises the recognized features' complexity because each layer combines

previous characteristics. As an example, the corner of a square is a higher level feature than vertical and horizontal lines because to detect it the combination of lines must be performed (the joint of them ensures a corner).

As shown in Figure 3.10, I used a CNN with a similar design as in [38], composed of 10 layers: 5 convolution layers (together with a ReLU unit), 4 pooling layers and a fully-connected layer with 400 hidden neurons and 33 outputs. In the first convolution layer I defined 60 activation maps. The next layers will increase the number of feature maps to finally get a representative feature vector of the image with 400 elements. Usually, in CNNs the convolution kernels are of size 3x3 or 5x5, so I used a combination of them with a pooling window size of 2x2. The parameters of each layer can be seen in Table 3.1. Each layer's output is the input for the next layer. The results in Section 4.5 showed that the depth and the feature numbers in each layer of this CNN are sufficient enough to solve the problem of recognizing SL letters and number symbols.

	Type	**Parameters**	Output
Layer 1	Convolution	Size: 5x5, Stride: 1	60 feature maps
Layer 2	Pooling	Size: 2x2, Stride: 2	60 feature maps
Layer 3	Convolution	Size: 5x5, Stride: 1	100 feature maps
Layer 4	Pooling	Size: 2x2, Stride: 2	100 feature maps
Layer 5	Convolution	Size: 3x3, Stride: 1	150 feature maps
Layer 6	Pooling	Size: 2x2, Stride: 2	150 feature maps
Layer 7	Convolution	Size: 5x5, Stride: 1	200 feature maps
Layer 8	Pooling	Size: 2x2, Stride: 2	200 feature maps
Layer 9	Convolution	Size: 3x3, Stride: 1	200 feature maps
Layer 10	Fully connected	Hidden neurons: 400	33 labels

Table 3.1: CNN layer description.

When an image is used as an input, to determine its corresponding American SL

symbol, it has to traverse all layers to activate output neurons in the fully connected layer. As shown in Figure 3.10, arrangement of the network layers is sequential, and thus feature maps computed in the previous layer are inputs of the actual layer. The last two layers are the accuracy and softmax layers, which are used to measure the error of predicted output and the real one to perform the tuning of the network.

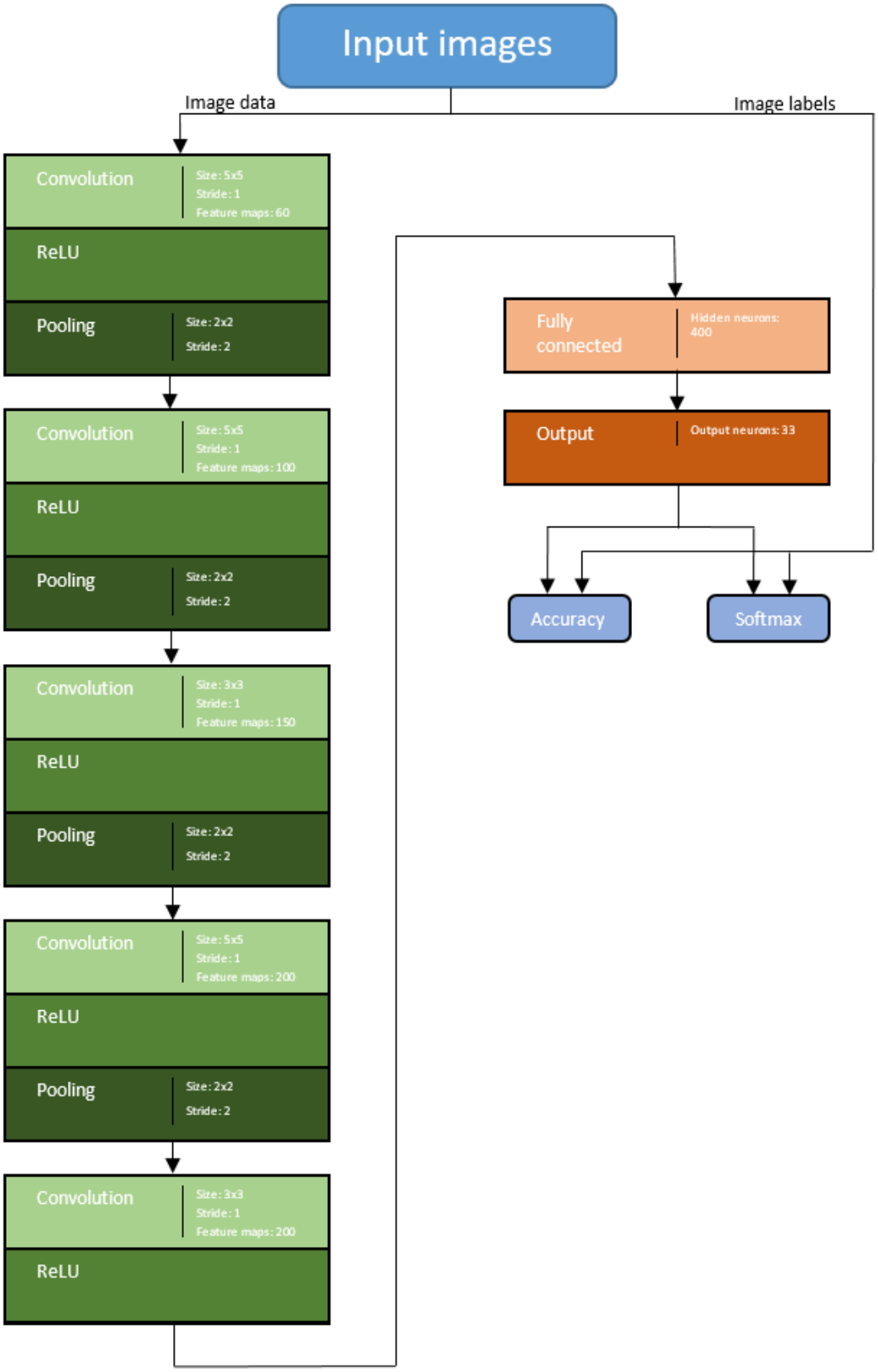


Figure 3.10: Architecture of the CNN.

CHAPTER 4

CAPTURE ENVIRONMENT AND EVALUATION

4.1 Ideal dataset

While I was performing some of the experiments with a Convolutional Neural Network (CNN) using the dataset presented in [67], I got perfect accuracies (100% of precision). I realized that the model may be overfitted. Viewing several images from the dataset I understood why I got these results: the images were similar and it seemed the images were adjacent frames of a video. As I stated in Section 3.2.3, I was not going to work with consequent video frames. Therefore, I searched for more datasets over the internet but the number of images was low and the focus of images was the entire person instead of just hands. Because of that, I decided to capture my own dataset gathering volunteer subjects from the American Sign Language department at Boise State University.

Images in the dataset were square regions of 100x100 pixels from the depth images. The cropping was performed based on user's body joint information: the hand was constantly being tracked until the user completed the required gesture. The extracted region was processed to normalize the values of the hand region, guaranteeing maximum variance.

The target audience of the dataset participants could be any person capable of performing gestures of American Sign Language (ASL). I captured volunteers from the American Sign Language department at Boise State University to improve the accuracy of data. Since students are still learning, gestures may not be totally correct but this provided variation in observations extending the covering of the classification model.

4.1.1 Ideal world definition

Depending on the complexity of the scene (colors, reflective materials, and other objects), a capture may not be good enough to detect, recognize or classify signs. Creating an "ideal room was a must to achieve good results for Hand Gesture Recognition system, see Figure 4.1. This ideal scene should not have anything that inteferes with the capture from the Kinect. When capturing each image I used the camera's depth data. This sensor had to produce images in which the contents are clear in order to be able to segment the hand.

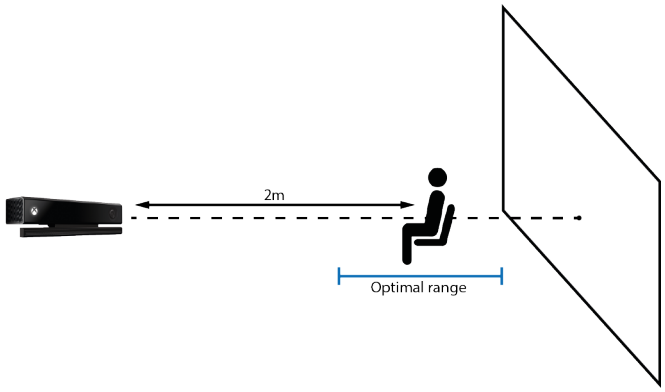


Figure 4.1: Ideal world scene.

Since the depth sensor was a light coding sensor the projected light may be reflected differently on different materials. This was a problem because the sensor measures depths wrong for pixels falling in these materials. Eliminating objects in the scene and having a smooth non-reflective background avoided these problems.

The user's position in the scene was important because the depth sensor had a limited effective range of capture. This effective range was between 1.5m and 3.5m. I

set the user's distance at 2m and placed them in the center of the field of view of the camera. The user should be able to move their hands with freedom when performing any sign. Since gestures are done with the upper half of the body, I captured users seated because they fit correctly and it was more comfortable for them.

4.1.2 Target images

The dataset had to contain images of users sitting in front of the camera performing Sign Language. Images must be captured when the user had performed the symbol; no intermediate gestures were allowed for capture. There must be different users to ensure variations of size and position of gestures in the images.

The main focus of capturing were users' hands. When the gesture was performed the hand must be visible and trackable. Because I focused on hand letter gesture recognition, hands must be facing towards the camera. Depending on the gesture, fingers may be occluded by the hand. Since in these type of cases occlusions are the characteristics of the gesture they were allowed. If the gesture does not contain finger or palm occlusions in its ideal form, occlusions were not allowed.

The goal of the dataset was to provide gestures in Sign Language for letters of the alphabet from A to Z and symbols for numbers. Sign Language has diverse languages and symbols for letters may vary or may add new ones to the alphabet. In this case, the selected language was American Sign Language. The inclusion of the numbers should add more gestures, and thus variety for the database, increasing the scalability of the problem.

As part of future work goals, replacing letters with hand gesture symbols to

represent objects (as it is usually done in sign language) tests the robustness of the system. Training models with different datasets and achieving high accuracies will show that the system is valid and robust in different applications. In the near future, I should be able to combine letters with symbols and numbers to get a bigger dataset. This will show if the system is robust enough to deal with the scalability number of gestures in Sign Language recognition problem.

4.1.3 Values of the images

Normal cameras capture color information of the scene but are susceptible to lighting conditions. Furthermore, using only colors may not be enough to detect the exact positions of fingers and thus the correct hand gesture. Since sensors of cameras which capture depth images of scenes are not susceptible to lighting conditions, they ensure that all gestures are captured in the same conditions. Information of depth scene captures the regions of objects and their depth. Therefore when capturing hands, distances from the camera to fingers and hand palm are known.

Since Kinect provides depth values within a range of [0-8000] (each unit represents one millimeter in the scene), its precision is high compared to other depth sensors. Capturing the hand in the scene by cropping a cube around the hand joint in depth space, allows the normalization of values of the hand region. The hand may be located at different distances from the camera and taking the sensor's raw data it is not suitable, as I stated in 3.2.3. Hand pixel values must be normalized into a certain range. In this case, I defined a range of 256 intensities, this is an 8-bit image.

4.1.4 Variations of observations

The dataset must not contain repeated images, all instances in it must be different. Using the classification model in the real world, users will perform differently than the dataset observations. To keep track of it, the dataset must have variations of the gestures. Rotating the gestures and shifting the hand over different locations of the scene ensures variance in the dataset. The rotations should not be too big, they should be enough to provide variance and still keep track of fingers and hand palm.

4.1.5 Size of the dataset

American Sign Language has 26 letters and 9 symbols to represent numbers. The letter J and the letter Z are done using finger movement and since I recognized static images, they were not captured. Therefore the number of letters and numbers remain in 33 symbols. The dataset must be balanced because if it is not, the dataset will be biased towards one specific class. The aim of the testing dataset is to capture 20 different users performing each symbol 10 times, resulting in a dataset of 6600 instances: 200 instances per symbol. This amount of gestures and variations of observations will allow the classification model to have diverse data for training and testing.

4.2 Captured dataset

I arranged the real scene to fit the ideal world requirements and I gathered the dataset as explained in Section 4.1. It is not possible to recreate ideal environments

for studying specific problems and altered the scene. In the previous section, I defined the ideal world for capturing images, but I realized I was going to need to change some characteristics of it. In this section, I explain changes I did, how I captured the images and the resulting images used to create the dataset.

This study was approved by the Human Subjects department at Boise State University, the Institutional Review Board (IRB) number is 131-SB17-015 [1]. The document approving the study is included in Appendix A The captured dataset involved 11 participants from which 2 were proficient, 1 was a fluent user and 8 were not fluent or did not know Sign Language (SL).

In Section 4.1 I defined the ideal arrangement of the scene and participants to capture images for the dataset. I explained how the distance from the camera to the user should be around 2m, but while I was performing the gathering I realized that hands in the images looked small compared to image size. This ensures room for the hand to move freely and provide variation in the dataset. Therefore I could decrease the size of images. Alternatively, some user's hands may be too small to have sufficient precision for capturing features. I decided to fix the image size and make users sit closer. Since the camera's field of view is a projection, objects close to camera are bigger than far ones. As I explained in Section 2.1, the Kinect has some problems capturing close objects but gathered data did not show any concerning problems.

While capturing images I noticed errors of the Kinect that appear in the entire set: most of the hands had brighter areas around the shape of them. Depending on the gesture, this area encloses the hand and the brightness of it changes from image to

[1]For further information about this approval it is possible to contact the Human Subjects department by email at humansubjects@boisestate.edu or by phone (208) 426-5401.

image. This means that the Kinect was capturing surrounding pixels as hand pixels. The brightness changes were due to normalization, if there was an area of the hand close to camera, the normalization adapts each pixel value to fit image pixel value range of [0-255] - this could be a further research work.

Expert users, users who knew SL or had practiced it for some time in their lives, tended to have their own accent. One of the participants from image gathering provided information about reasons for these accents: to sign faster and make faster transitions from one sign to the next one, they do not perform the whole sign. As an example, gestures for letters m and n are performed with the index and middle fingers closed, but for faster signing expert users keep them extended as shown in Figure 4.2. Another discrepancy between the different level of expertise in SL, was that nonexpert users tried to match symbols, and thus, variations of gestures were not as large as with expert users. Their captured images have more variations in hand location in the image than the expert users because these last ones tried to be still with more gesture variations for the same symbol.

Participants were told not to try to be still while taking images and to rest between captures. Because of that, captured images had variations over x and y axises of images. Since they rested after each capture when they engage to get the next shot, the depth value of hand (z axis) were different providing more variation to the dataset. These variations made hands appear in images in different locations and with diverse sizes. Depth values were normalized as explained in Section 3.3.2, therefore all the values were mapped from [0-8000] to [0-255].

The resulting dataset contained 3615 grayscale images of 100x100 pixels. From

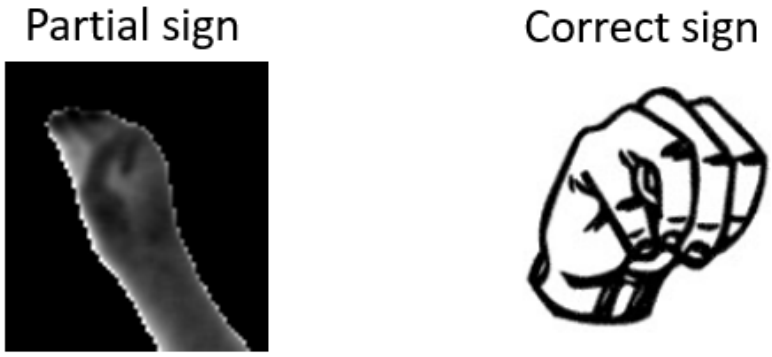


Figure 4.2: Left image is an intermediate gesture for faster signing, right image shows the correct symbol for m letter. Index and middle finger are extended to change to the next symbol in a faster way.

the total number of images, I split the dataset into training and testing images: 2890 had been selected for train set and 725 for test set.

Taking a look at captured images, since the minimum value for the hand is mapped as 0, they tended to be dark, especially when the gesture pointed towards the camera. Fingers extended and pointing to the camera created dark areas but these pixel intensity variations were features of the gesture. These features were the ones that the CNN had to learn to be able to recognize each gesture.

4.3 Used datasets

For this study I used two datasets: the first one is a Vietnamese dataset from the work presented in [67] and the second one is an American Sign Language dataset I collected for this study. I selected the Vietnamese dataset because it fits my requirements for the problem: images were enclosing only hands with no other object nor background.

The number of images in this database was enough to generalize diverse gestures, furthermore, in Vietnamese SL there are special accents for some letters and special gestures for double letters. The images in this set were *.png* images, and thus, compared to other datasets I was able to see the content of them.

I did not find any American Sign Language dataset which fit my requirements. The ones I found were color images with backgrounds or depth images with the body of the participant (raw captures of the Kinect). For this work, the constraint was to have only images containing hands of the users. Because of that, I captured the dataset defined in Section 4.2 and I used it to train and test the CNN, together with the image *average* technique.

4.4 Evaluation method

I used the image average technique for comparing images in a dataset. This method was based on measuring differences of images. When an input image was fed to the system the class in the database that had the least average error was the recognized gesture [64].

If all images of each class in the dataset are similar, this technique achieves high accuracy because image difference is minimal. In the "perfect" case: when all images in each class are the same, when comparing the input image the difference of each pixel is 0 and thus the average error is 0. Adding more variations (different images) to the dataset, the error increases, and thus, with high variance, the method fails.

To evaluate the variance of the datasets and the generalization they provide for training purposes, I used the image average technique. If this method achieves high

accuracy it means that images for training are similar, consequently, the variation the dataset offers is limited. Otherwise, if the technique performs poorly it may be because of two reasons: (1) the dataset is not correct and images are not correctly labeled, or (2) images for each class have high variations to represent each gesture [64, 18].

To evaluate the CNN I split the collected dataset into training and testing sets, as I explained in Section 4.2, taking 2890 images for training and 725 for testing. For the Vietnamese dataset, I randomly split the database into 4106 images for training and 330 for testing.

I used the standard method used in the Caffe library for training. For the CNN validation phase, I used 200 mini batches with a test interval of 500 iterations. I set the learning rate of the network at 1×10^{-5} and after each test interval, it decreases by a factor of 0.1, while initialization of weights is based on *xavier* initialization. With this set up for the backpropagation algorithm of the CNN, the network reaches a suitable loss in 6000 iterations and achieves a 91.25% of accuracy over the training set with 0.1661 loss. If the precision reaches 100% in the training phase, it means the CNN is overfitted and that I need a bigger dataset or reduced network size.

Usually, the precision of a system is computed using the best matching label in the dataset, but sometimes allowing some flexibility achieves better results [38]. Taking a look at collected images I realized some of the gestures looked similar even though they were different symbols. The system may have problems distinguishing those images to assign the correct label, and thus the accuracy of the system decreases. In [38] the method of evaluating the CNN was by measuring the error rate using top-1

and top-5 predictions. Using this approach, if the possible classifications classes were similar, it was possible to know if the system was close to predicting the correct label. Since the number of classes in this problem was lower than the used ones in [38], I tested the results overall top-k outputs. I used a different number of users to see if the tendency of the accuracies was kept. If the tendency was not linear and had a logarithmic shape, it meant there existed a k threshold for top-k which after that point I could not get further meaningful information.

4.5 Results

After processing data with these methods and with the evaluation strategy, I got promising results. The obtained results over the Vietnamese dataset are shown in Figure 4.3: with both approaches I acquired high accuracy. Only taking the best matching prediction (top-1) I got 97% of correct assigned labels, and if I search for the correct label in the top-3 I reach up to 100%. In Figure 4.4 the image class *average* method and the CNN results are shown for a different number of users over the captured dataset: in this case 1, 5 and 11 users. The test with the entire dataset (11 users) showed the best performance because of the variation of the data; the previous two tests were done with nearly overfitted networks due to low amount of data.

Results showed in Figure 4.4 how the image *average* method did not learn features from images. Since the tendency of the top-k was linear and the first top results had low precision, the method was not useful for this problem. The collected dataset had high variance compared to Vietnamese dataset and that was why it performed worse.

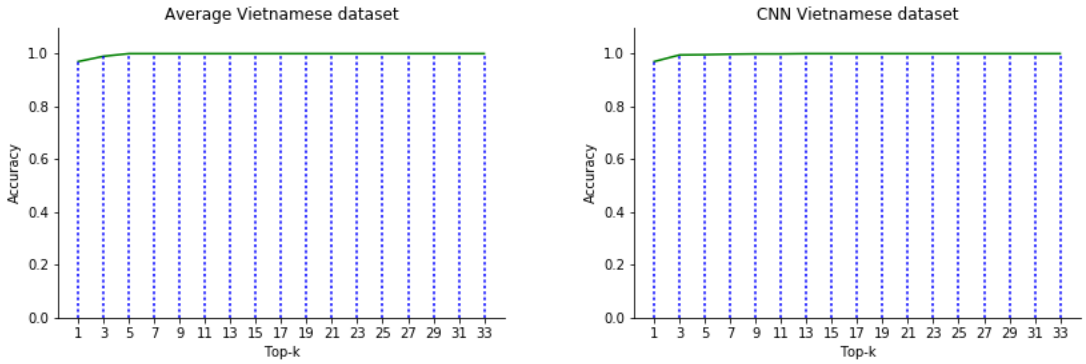


Figure 4.3: Results over the Vietnamese dataset.

When I got these results for Vietnamese data I took a deeper look at the images and realized that low variance in hand positions was due to the fact that the images the dataset were consecutive frames of a video.

The CNN results over the captured dataset showed logarithmic tendency which meant it learned characteristics of hand images. This tendency implied that there was a threshold k for top-k which, for higher values of k, there was no meaningful information. In Figure 4.4, I highlighted in red the first threshold which provides an accuracy above 75%. For this problem, this precision is enough to be able to classify correctly without having any overfitting [38, 41].

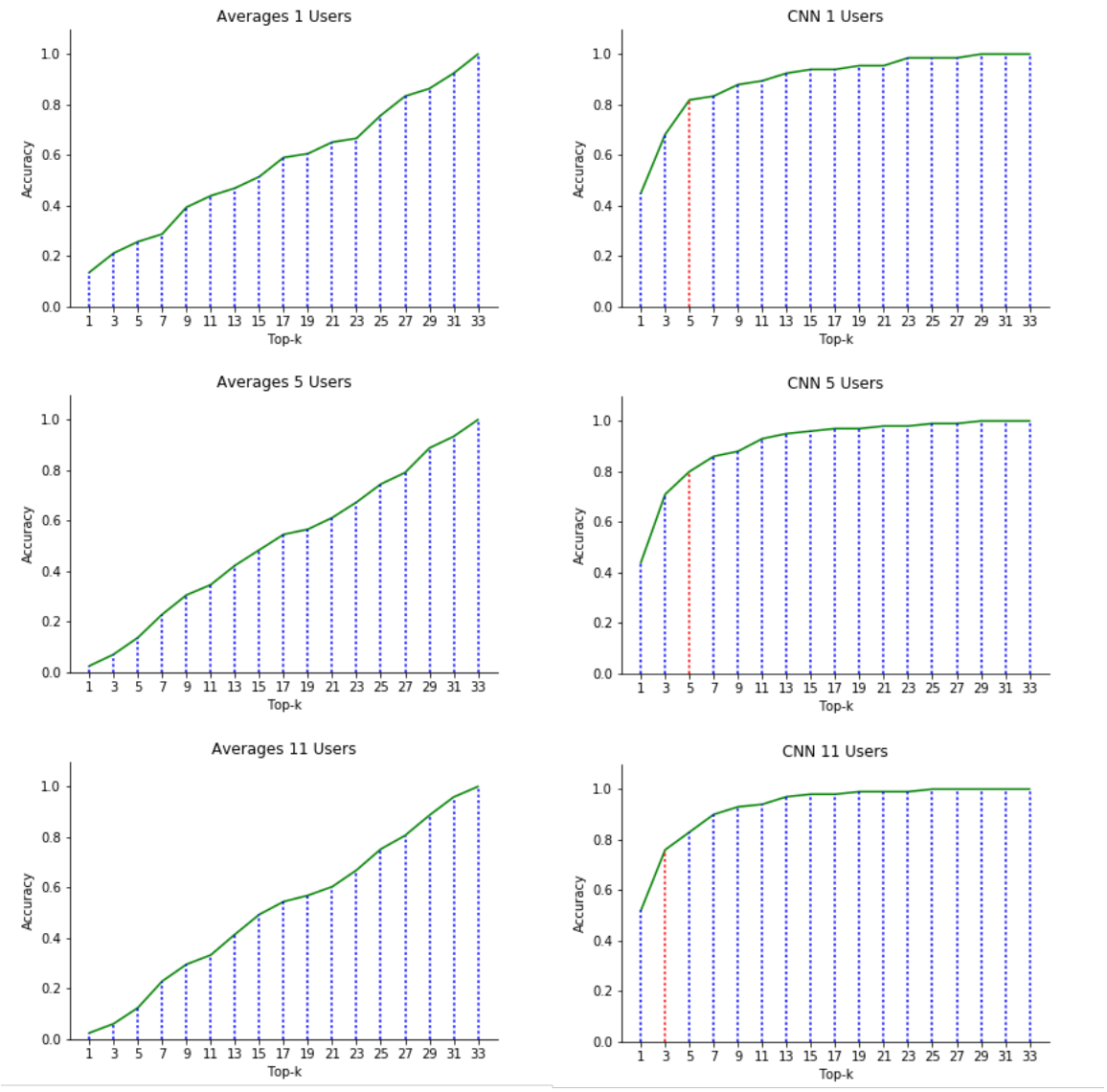


Figure 4.4: Results over the collected dataset.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The problem of Sign Language (SL) recognition using images is still a challenge. Similarity of gestures, user's accent, context and signs with multiple meanings lead to ambiguity. These are some reasons why previous work used limited datasets. In this manuscript, I used Random Decision Forests (RDF) and Neural Networks (NN) to conduct a preliminary study about SL recognition using depth images. This study provided information about the constraints of the problem. I concluded that a dataset to train and evaluate the system must have sufficient gesture variations to generalize each symbol. I extended the *average* method to determine if a dataset has enough gesture variations and I implemented a Convolutional Neural Network (CNN) for recognizing hand gestures in images of American Sign Language letter and number symbols. The *average* method was based on [64], but in my case the focus was hand images and I parallelized the method using the GPU to increase the efficiency.

For conducting experiments, I only found one external dataset (Vietnamese Sign Language letters) which fulfilled my three requirements: (1) images must be depth images, (2) the hand should be totally visible, and (3) background and foreground removed. Most of the SL datasets are color images or depth images with no pre-processing. These images had background and users' bodies. Considering there was not an "ideal" American Sign Language dataset, I gathered images to create a new dataset. There was a big difference between the two datasets: gesture shape variations in the Vietnamese dataset was low while American dataset was created focusing on providing high variance. Results demonstrated how the developed method worked to show low variance in the Vietnamese dataset, achieving high recognition rate, and

high variance of the American dataset, achieving low recognition rate.

I evaluated the system with two different datasets: hand images from Vietnamese Sign Language letters and from American Sign Language letters and numbers. SL can have a lot of variations for each gesture; accent of a person, signing speed, and the next symbol determines gesture shape. Therefore, a dataset has to have enough variations of each gesture to capture most of the possible shapes a gesture can have. I developed a classification method based on a previous work [64] for face recognition to detect whether or not a dataset provides enough gesture shapes. Since the method creates a representation of each gesture, if it acquires high recognition rate, the dataset has similar images, which is not good to generalize the problem of SL recognition. If the method cannot recognize gestures, the dataset will have enough variance to provide information about diverse shapes of the gestures.

Compared to the *average* method, the CNN achieved a higher recognition rate. Because American Sign Language has very similar symbols (m and n for example) for some of the letters and gesture shape variation in the dataset is high, the system had problems to classify correctly some of the images. Some of the images looked similar and the CNN labeled them incorrectly. Instead of taking the best classification (top-1), if in the top-3 possible labels was the correct one, it meant the system was close to recognizing the image, but due to similar symbols the CNN missclassificated it. Looking further than top-5, the CNN did not provide meaningful information, suggesting top-5 classifications were enough to achieve high recognition rate.

The collected dataset of American Sign Language letters and symbols is going to be shared with the scientific community. It should be extended to more fluent

and proficient users because the aim of this work is for deaf persons who are fluent users. However, collected non-fluent user images was greater than the number of fluent users, and thus, images may not fulfill the requirements of those users. Due to the lack of dexterity in hands of non-fluent users, the captured images may not fit gesture shapes of fluent users and thus the created system would not achieve high accuracy recognition rate for deaf people.

5.2 Future work

At this moment, results showed high accuracy on the captured dataset with symbols for letters and numbers, but the CNN has been designed based on my own experience with some tests and tutorials I did before and following the architecture presented in [38]. It may be that adding more symbol classes to recognize, the model precision decreases and thus the system will not be as good as before. My idea to improve system's design is to start with a group of gestures and test diverse configurations of the CNN to observe which configurations are the best for that case. The group of gestures will be increased with additional classes and, using the evaluation methods, identify the best configuration. From the relation number of classes and best configurations, my goal is to find a relation between input number of classes and design of the CNN so the system can construct an effective model for each case.

Once the letter recognition system has achieved high recognition rate, a logical step towards the SL transcription is the concatenation of predicted letters to construct words. Since the CNN for letter recognition is not perfectly precise, errors exist at the recognition step, consequently, errors can accumulate over the transcription

process. My current knowledge about language processing is not high enough to define exact features of this future work but one idea is to use Hidden Markov Models for the creation of finger-spelled words and correct errors caused by the CNN. SL has symbols to represent words instead of finger-spelling them, therefore if a word gesture is detected the word creation step is skipped. Recognized words will then be used to construct sentences in the final step of the transcription process. SL has no direct translation to natural language due to the elusion of some of the words; as an example, stop words or prepositions. Techniques used in query suggestion systems may be useful to ensure correctness of sentence forming, but they have to be trained in already transcripted SL texts.

As stated in Section 4.1, the proposed system was trained only using static images; such a case excludes some of the letters in American Sign Language alphabet (J and Z). The inclusion of these letters and gestures done by movement is crucial for proper transcription, thus methods to recognize gestures will be required in the future.

In Section 2.4, I mentioned the significance of face expressions. I would like to add this feature to the system because it may be helpful for understanding what the users are saying, and thus produce more precise transcriptions. The use of another CNN should be useful to detect face expressions, that is one area I would like to study to improve the transcription system.

REFERENCES

[1] Cuda dnn - cudnn. https://developer.nvidia.com/cudnn.

[2] Cyber science. https://apps.leapmotion.com/apps/cyber-science-motion/windows.

[3] Galaxy zoo - the galaxy challenge. https://www.kaggle.com/c/galaxy-zoo-the-galaxy-challenge.

[4] Large scale visual recognition challenge 2010. http://image-net.org/challenges/LSVRC/2010/.

[5] Large scale visual recognition challenge 2015. http://image-net.org/challenges/LSVRC/2015/index.

[6] Leap motion. https://www.leapmotion.com/.

[7] Shortcuts. https://apps.leapmotion.com/apps/shortcuts/windows.

[8] Structured light. https://en.wikipedia.org/wiki/Structured_light.

[9] Time-of-flight. https://en.wikipedia.org/wiki/Time_of_flight.

[10] Time-of-flight. https://en.wikipedia.org/wiki/Stereo_camera.

[11] Tiny dnn. http://tiny-dnn.readthedocs.io/.

[12] Ossama Abdel-Hamid, Abdel-Rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[13] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*, pages 4277–4280. IEEE, 2012.

[14] Anant Agarwal and Manish K Thakur. Sign language recognition using microsoft kinect. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 181–185. IEEE, 2013.

[15] Mohamed N Ahmed and Aly A Farag. Two-stage neural network for volume segmentation of medical images. *Pattern Recognition Letters*, 18(11):1143–1151, 1997.

[16] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Consumer Depth Cameras for Computer Vision*, pages 71–98. Springer, 2013.

[17] Adam Baumberg. Reliable feature matching across widely separated views. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 774–781. IEEE, 2000.

[18] Roberto Brunelli and Tomaso Poggio. Face recognition: Features versus templates. *IEEE transactions on pattern analysis and machine intelligence*, 15(10):1042–1052, 1993.

[19] Francesco Camastra and Domenico De Felice. Lvq-based hand gesture recognition using a data glove. In *Neural Nets and Surroundings*, pages 159–168. Springer, 2013.

[20] Chun-Te Chu, Kuan-Hui Lee, and Jenq-Neng Hwang. Self-organized and scalable camera networks for systematic human tracking across nonoverlapping cameras. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2322–2326. IEEE, 2013.

[21] Jon Coaffee. Rings of steel, rings of concrete and rings of confidence: designing out terrorism in central london pre and post september 11th. *International Journal of Urban and Regional Research*, 28(1):201–211, 2004.

[22] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[24] Cícero Nogueira dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78, 2014.

[25] Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networksa review. *Pattern recognition*, 35(10):2279–2301, 2002.

[26] David B Fogel, Eugene C Wasson, Edward M Boughton, and Vincent W Porto. A step toward computer-assisted mammography using evolutionary programming and neural networks. *Cancer letters*, 119(1):93–97, 1997.

[27] Rafael C Gonzalez and Richard E Woods. Digital image processing, 2008.

[28] 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.

[29] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[30] Yusuke Imamura, Shingo Okamoto, and Jae Hoon Lee. Human tracking by a multi-rotor drone using hog features and linear svm on images captured by a monocular camera. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2016.

[31] Shoichiro Iwasawa, Kazuyuki Ebihara, Jun Ohya, and Shigeo Morishima. Real-time estimation of human body posture from monocular thermal images. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 15–20. IEEE, 1997.

[32] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[33] Nebojsa Jojic, Barry Brumitt, Brian Meyers, Steve Harris, and Thomas Huang. Detection and estimation of pointing gestures in dense disparity maps. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 468–475. IEEE, 2000.

[34] Maria Karam. *A framework for research and design of gesture-based human computer interactions*. PhD thesis, University of Southampton, 2006.

[35] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[36] Nidhi N Khatri, Zankhana H Shah, and Samip A Patel. Facial expression recognition: A survey. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 5(1):149–152, 2014.

[37] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] 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.

[40] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273, 2015.

[41] 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.

[42] Sung Chun Lee and Ram Nevatia. Hierarchical abnormal event detection by real time and semi-real time multi-tasking video surveillance system. *Machine vision and applications*, 25(1):133–143, 2014.

[43] Gil Levi and Tal Hassner. Age and gender classification using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 34–42, 2015.

[44] Shuai-Shi Liu, Yan-Tao Tian, and Dong Li. New research advances of facial expression recognition. In *Machine Learning and Cybernetics, 2009 International Conference on*, volume 2, pages 1150–1155. IEEE, 2009.

[45] Shih-Chung B Lo, Heang-Ping Chan, Jyh-Shyan Lin, Huai Li, Matthew T Freedman, and Seong K Mun. Artificial convolution neural network for medical image pattern recognition. *Neural networks*, 8(7):1201–1214, 1995.

[46] Liying Ma, Yegui Xiao, Khashayar Khorasani, and Rabab Kreidieh Ward. A new facial expression recognition technique using 2d dct and k-means algorithm. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 2, pages 1269–1272. IEEE, 2004.

[47] Sotiris Malassiotis, Niki Aifanti, and Michael G Strintzis. A gesture recognition system using 3d data. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 190–193. IEEE, 2002.

[48] Hayato Mano, Kazuyuki Kon, Noritaka Sato, Masataka Ito, Hisashi Mizumoto, Kiyohiro Goto, Ranajit Chatterjee, and Fumitoshi Matsuno. Treaded control system for rescue robots in indoor environment. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1836–1843. IEEE, 2009.

[49] 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.

[50] Javier Molina, Marcos Escudero-Viñolo, Alessandro Signoriello, Montse Pardàs, Christian Ferrán, Jesús Bescós, Ferran Marqués, and José M Martínez. Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models. *Machine vision and applications*, 24(1):187–204, 2013.

[51] 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.

[52] 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.

[53] Leigh Ellen Potter, Jake Araullo, and Lewis Carter. The leap motion controller: a view on sign language. In *Proceedings of the 25th Australian computer-human*

interaction conference: augmentation, application, innovation, collaboration, pages 175–178. ACM, 2013.

[54] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 246–253. Springer, 2013.

[55] 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.

[56] 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.

[57] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[58] Berkman Sahiner, Heang-Ping Chan, Nicholas Petrick, Datong Wei, Mark A Helvie, Dorit D Adler, and Mitchell M Goodsitt. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE transactions on Medical Imaging*, 15(5):598–610, 1996.

[59] 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.

[60] Mel Slater, Bernhard Spanlang, Maria V Sanchez-Vives, and Olaf Blanke. First person experience of body transfer in virtual reality. *PloS one*, 5(5):e10564, 2010.

[61] Luuk J Spreeuwers, AJ Hendrikse, and KJ Gerritsen. Evaluation of automatic face recognition for automatic border control on actual data recorded of travellers at schiphol airport. In *Biometrics Special Interest Group (BIOSIG), 2012 BIOSIG-Proceedings of the International Conference of the*, pages 1–6. IEEE, 2012.

[62] Ruofeng Tong, Di Xie, and Min Tang. Upper body human detection and segmentation in low contrast video. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(9):1502–1509, 2013.

[63] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[64] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[65] Dominique Uebersax, Juergen Gall, Michael Van den Bergh, and Luc Van Gool. Real-time sign language letter and word recognition from depth data. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 383–390. IEEE, 2011.

[66] Paul Viola, Michael J Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[67] Duc-Hoang Vo, Trong-Nguyen Nguyen, Huu-Hung Huynh, and Jean Meunier. Recognizing vietnamese sign language based on rank matrix and alphabetic rules. In *Advanced Technologies for Communications (ATC), 2015 International Conference on*, pages 279–284. IEEE, 2015.

[68] Yong Wang, Tianli Yu, Larry Shi, and Zhu Li. Using human body gestures as inputs for gaming via depth analysis. In *2008 IEEE International Conference on Multimedia and Expo*, pages 993–996. IEEE, 2008.

[69] Renliang Weng, Jiwen Lu, Junlin Hu, Gao Yang, and Yap-Peng Tan. Robust feature set matching for partial face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 601–608, 2013.

[70] Paul Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.

[71] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[72] Yegui Xiao, Liying Ma, and Khashayar Khorasani. A new facial expression recognition technique using 2-d dct and neural networks based decision tree. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 2421–2428. IEEE, 2006.

[73] Mubashira Zaman, Soweba Rahman, Tooba Rafique, Filza Ali, and Muhammad Usman Akram. Hand gesture recognition using color markers. In *International Conference on Hybrid Intelligent Systems*, pages 1–10. Springer, 2016.

[74] Hao Zhang, Christopher Reardon, and Lynne E Parker. Real-time multiple human perception with color-depth cameras on a mobile robot. *IEEE transactions on cybernetics*, 43(5):1429–1441, 2013.

[75] Wenlu Zhang, Rongjian Li, Houtao Deng, Li Wang, Weili Lin, Shuiwang Ji, and Dinggang Shen. Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, 108:214–224, 2015.

[76] Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning identity-preserving face space. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 113–120, 2013.

[77] Thi Thi Zin, Hideya Takahashi, and Hiromitsu Hama. Robust person detection using far infrared camera for image fusion. In *Innovative Computing, Information and Control, 2007. ICICIC'07. Second International Conference on*, pages 310–310. IEEE, 2007.

APPENDIX A

IRB APPROVAL

BOISE STATE UNIVERSITY
RESEARCH AND ECONOMIC DEVELOPMENT

Date: February 09, 2017

To: Iker Vazquez Lopez cc: Steven Cutchin

From: Office of Research Compliance (ORC)

Subject: SB-IRB Notification of Exemption - 131-SB17-015

Hand Gesture Recognition For Sign Language Transcription Using Depth Cameras

The Boise State University ORC has reviewed your protocol application and has determined that your research is exempt from further IRB review and supervision under 45 CFR 46.101(b).

Protocol Number: 131-SB17-015

Approved: 2/9/2017 Application Received: 2/1/2017

Review: Exempt

Category: 2

This exemption covers any research and data collected under your protocol as of the date of approval indicated above, unless terminated in writing by you, the Principal Investigator, or the Boise State University IRB. All amendments or changes (including personnel changes) to your approved protocol **must** be brought to the attention of the Office of Research Compliance for review and approval before they occur, as these modifications may change your exempt status. Complete and submit a Modification Form indicating any changes to your project.

Annual renewals are not required for exempt protocols. When the research project is completed, please notify our office by submitting a Final Report. The exempt status expires when the research project is completed (closed) or when the review category changes as described above.

All forms are available on the ORC website at http://goo.gl/D2FYTV

Please direct any questions or concerns to ORC at 426-5401 or humansubjects@boisestate.edu.

Thank you and good luck with your research.

Office of Research Compliance