Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Bachelor's thesis

GESTURE RECOGNITION LIBRARY FOR LEAP MOTION CONTROLLER

Michał Nowicki, 95883
Olgierd Pilarczyk, 100449
Jakub Wąsikowski, 101560
Katarzyna Zjawin, 98826

Supervisor
dr inż. Wojciech Jaśkowski

Poznań, 2014

Tutaj przychodzi karta pracy dyplomowej;
oryginał wstawiamy do wersji dla archiwum PP, w pozostałych kopiach wstawiamy ksero.

Contents

Chapter 1

Abstract

Nowadays, when computers are ubiquitous, people need to develop more natural and intuitive interface to use computer. People would like to use gestures of everyday life and translate them into a virtual world. This paper presents library for gesture recognition dedicated to Leap Motion Controller called LMGesture. Leap Motion is a new device, which tracks fingers and other objects up to 1/100th of a millimeter. The LMGesture library can be used for various kinds of gestures. This can be done, because of the use of different detection methods for different types of gestures. For static gestures recognition has been used support vector machine (SVM) and for dynamic gestures – hidden Markov model (HMM). In library can be found additional modules: recorder – for recording gestures in a format supported by the library, visualizer – for reviewing recorded gestures, finger differentiation module – for differentiating fingers in a performed gesture. In this paper has been also presented classification of gestures and its modification in the context of Leap Motion Controller. Additionally this work includes descriptions of helper methods used for the gesture recognition, such as pre-processing of data obtained from the device, which gets rid of existing noise or method for fingers differentiating, whereby the obtained results are more accurate.

[dodac opis testow]

Chapter 2

Introduction

2.1 Scope of work

The objective scope of this paper includes:

- designing library architecture,
- selection and implementation of algorithms for gesture recognition,
- implementation (or learning) of gestures built-in library,
- implementation examples of the library,
- tests using Leap Motion Controller.

Following thesis concerns machine learning. In this paper were used two algorithms from this field of science: support vector machine and hidden Markov model. SVM is included in the group of supervised learning, where algorithm knows set of input data and responses to the data, and tries to create a predictor model that generates reasonable predictions for the response to new data. HMM is an example of unsupervised learning, where algorithms are trying to find hidden structure using unlabeled data.

Subjective scope of this work is to examine Leap Motion Controller in gesture recognition. This device is an innovative approach to the computer usage. The scope of this work is to examine this controller and create an interface between it and the user.

Time range of the thesis is October 2013 – January 2014.

2.2 Motivation

Nowadays computer usage is not a natural human behavior. To rotate the object in the virtual world user need to click the mouse and move it in 2D plane. Man rotating objects in the real world has to catch it and turn using hands. To make the usage of computer more intuitive and natural the best solution would be to transfer gestures performed on a daily basis into the virtual world. The latest technological solutions allow to control computer using gestures. Hands are a fundamental tools of every human being. With them people perform hundreds of operations every day. Without this basic manipulator people are not able to cope with the simplest activities. Hands give a large scope of activities and gestures. They are even used for non-verbal communication. Using this tool man at a low cost can do and achieve almost everything. Currently there are several devices that support gesture recognition. An example of such a controller is Kinect, but it is not highly accurate. It is suitable for applications that use the whole body of the user, but for the recognition

of hand gestures may be no useful. Leap Motion Controller is a small device that can be placed in front of the computer. Its operating range is between user and PC, and its accuracy is very high. It is an ideal device for recognizing hand gestures through which people have the opportunity for more natural user interface of the computer.

2.3 Objectives

The main purpose of this paper is creation of library, which recognizes hand gestures using Leap Motion controller. Additionally,below bulleted objectives will also be realized:

- comparison and evaluation of existing methods in the context of hand gesture recognition,
- creating a module to manage gestures,
- creating an initial database of gestures to the library,
- performing hand gesture recognition quality tests,
- presentation of example of library usage.

2.4 Thesis organization

The structure of the paper is as follows. Chapter 5 presents an overview of the literature on gesture recognition. In this part there are descriptions of gestures classification and known methods of gesture recognition. Chapter 6 is a presentation of Leap Motion Controller. Chapter 7 is devoted to gestures recognizing in the context of Leap Motion Controller. There are descriptions of gestures classification, data representation and additional processing steps for hand gestures recognition using the Leap Motion device. Chapter 8 contains proposed methods, evaluation methodology and experiments for static gesture recognition. Chapter 9 presents a similar description for dynamic gesture recognition. In chapter 10 has been described created library - its architecture, processes, additional modules, used libraries for its implementation and an example of its usage. Chapter 11 provides conclusions of the paper.

Michał Nowicki - svm do statycznych, hmm do dynamicznych Olgierd Pilarczyk - projektowanie architektury, impementacja preprocessingu Jakub Wąsikowski - visualizer, recorder, rozróżnianie palców, wyznacznie liczby klastrów, klasyfikacja gestów Katarzyna Zjawin - visualizer, recorder, rozróżnianie palców, klasyfikacja gestów

Chapter 3

Introduction to gesture recognition

3.1 Classification of gestures proposed in literature

The vast multiplicity of gestures that human can perform makes the number of classes – to which we can divide these gestures – substantial. Therefore the classification can be performed in different ways, taking into account the different characteristics of gestures. Most of presented theories include the knowledge that originates from a variety of science such as anthropology, linguistics, cognitive science and other. In this chapter it is provided review of the most common gesture classifications in Human Computer Interaction (HCI) context. It was focused mainly on gestures that are relate to hand and arm movements.

The basic classification of gestures is the division into static and dynamic gestures. Group of static gesture includes fixed gestures which are not take into account the changes in time. Dynamic gestures is group of time varying gestures.

There is also another general division considered by Kammer et al. [24] due to the type of actions activated by the gesture, to online and offline. The first group includes gestures which are processed during performing. It is a group of direct manipulation gestures that provide additional information about dynamics of gesture. There are often used to manipulate objects in space. The second one is group of action gestures, which are processed at the end of gesture. Most often these are gestures that convey the occurrence of specific meaning.

Karam and schraefel [25] proposed more extensive gesture taxonomy dedicated for Human Computer Interaction. Their classification is based on Quek et al. [36] publication, which provides clarification of gestures taxonomies presented in the past literature. Karam et al. defined following gesture classes: deictic, gesticulation, manipulation, semaphores and sign language. In another publication, Aigner et al. [41] presented classification, which are more tailored for hand gesture recognition purposes, drawing on concepts from Karam and schraefel work. They distinguished five categories of gestures:

- pointing – used to point object or indicate direction. Formally, it involve pointing to establish the identity or spatial location of an object within the context of the application domain [25]. It applies not only an indication by the index finger but also to any finger and any number of fingers. It is also independent of finger orientation and curvature, while gesture has a indication meaning. Equivalent to deictic gesture in Karam and Schraefel literature.

- semaphoric – group which consists gesture posture and dynamics, which are used to convey specific meanings. Formally, Semaphoric approaches may be referred to as "communicative" in that gestures serve as a universe of symbols to be communicated to the machine [36].

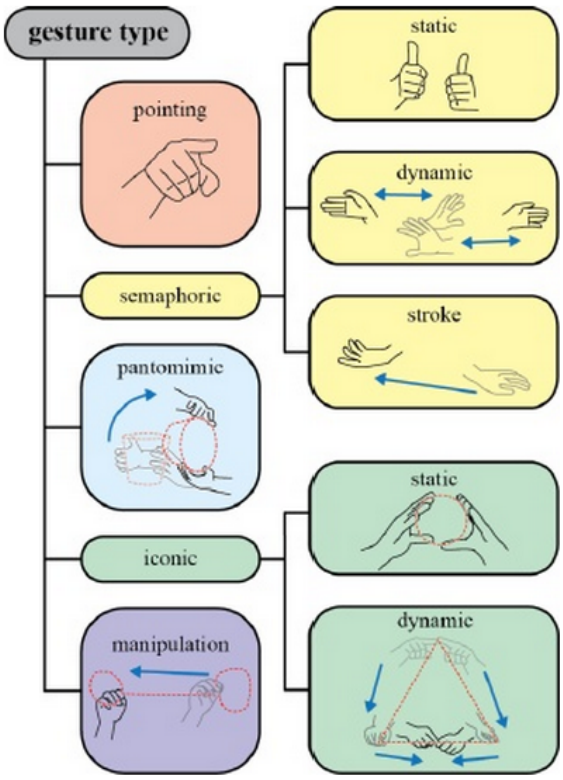

FIGURE 3.1: The classification of gestures proposed by Aigner et al. [41]

Due to the fact that semaphoric are symbolic gestures, their layout can be unrelated to their meaning. Distinguished three semaphoric gesture types of static, dynamic and strokes. The first one concerns specific hand posture, such as thumbs-up meaning approval symbol. Dynamic semaphorics convey their meaning through movement, for example waving of hand to greet somebody. The last one group are similar to dynamic semaphorics gesture, but this represents fast, stroke-like movements, such as swipe gesture.

- iconic – used to demonstrate shape, size, curvature of object or entities. In contrast to the semaphoric gestures, their layout or motion path is strictly related to their meaning. Iconic gestures can be divided into static and dynamic. First one are performed by hand postures, such as rectangle formed by the thumb and forefingers of both hands. Dynamic iconic gestures are often used to map edge line of objects by means of motion paths. For instance showing a simplified sine function characteristics with finger movements.

- pantomimic – presents imitated perform of specific task or activity without any tools or objects. Pantomimic gesture are characterized by a high variability of posture and movements. An example of this gesture type can be weapon reload or movement of a knife slicing bread.

- manipulation – used to control the position, rotation and scale of the object or entity in space. Manipulation gestures constitutes a direct interaction between the manipulated object and hand or tool that performs gesture. It follows, that the movement of the manipulated object must be strictly dependent on the motion gesture.

3.2 State of the art methods

In this chapter presented review of the state-of-the-art in human gesture recognition. The problem of gesture recognition can be divided in two main problems: the gesture representation problem and the decision/inference problem. Therefore, review includes discussion about enabling technology, gesture representations and analysis of recognition methods. Additionally introduced general problems related to the recognition of gestures and their common solutions.

3.2.1 Enabling Technology

In this subsection, overviewed the enabling technology for gesture recognition. The main existing gesture recognition approaches related to type of the devices are as follow:

- non-vision-based devices – tracking devices, instrumented gloves, armbands and others,
- vision-based devices – using one or many cameras.

Non-vision-based Technology

This type of devices uses various technologies to detect motions, such as accelerometers, multi-touch screens, EMG sensors and other, which includes several detectors. There are few categories of non-vision-based devices [23]:

- wearable – these kind of device is in the form of garment, which includes sensors needed to recognize arrangement and motions of examined part of body. Often occur in the form of gloves (CyberGlove®), armband (Myo) or the whole outfit (IGS-190). For instance, CyberGlove® device was used in system developed by Kevin et al. [26], which recognize multi-dimensional gestures using condensation-based trajectory matching algorithm. These devices are often related to biomechanical and inertial technologies,
- biomechanical – type of device, which use biomechanical techniques such as electromyography, to measure parameters of gesture. Example of using this type of device is project developed by Kim et al. [27] for Realtime Biosignal Interfacing based on EMG sensors. Example of these devices is Myo armband, which detects gestures and movements using EMG sensors,
- inertial – these devices measure the variation of the earth magnetic field in order to detect the motion. This kind of devices uses accelerometers [29] and gyroscopes [20] to measurements,
- haptics – various kinds of touch screens. For instance, Webel et al. [50] developed module for dynamic gestures recognition in multi-touch devices,
- electromagnetic – these devices measure the variation of an artificial electromagnetic field deriving from wireless networks, electronic devices or produced by themselves. Example of such devices is WiSee, which leverages ongoing wireless transmissions in the environment (e.g., WiFi) to enable whole-home sensing and recognition of human gestures [35].

Vision-based Technology

Vision-based devices include one or several cameras and provide perfomed data from the captured video sequences. Processing of frame is based on filtering, analyzing and data interpreting. The following types of vision-based technology can be distinguished [23][51]:

- typical video cameras – gesture recognition techniques based on data derived from monocular camera using detection methods such as color or shape based techniques, learning detectors from pixel values or 3d model-based detection,
- stereocameras – techniques based on captured images from two cameras, which provide an approximation of the recorded data to a 3d model representation,
- active techniques – require the projection of some form of structured light. Examples of this kind devices are Kinect or Leap Motion,
- invasive techniques – systems which require using of body markers such as color gloves [48], LED lights (Play Station Move controller).

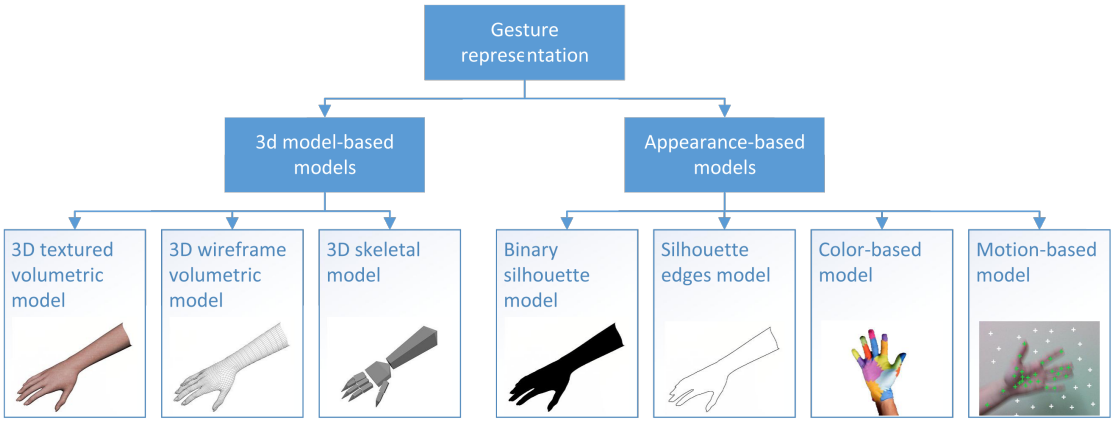

FIGURE 3.2: Diagram of gesture representation

3.2.2 Gesture representation

In this subsection, provides overview the spatial modeling of gestures. In particular, it was focused on one type of spatial gesture representations namely 3d model-based. Depending on the type of the input data, the approach for recognize gesture could be done in different ways. There are following two main types of gesture representation defined in literature [22][32]:

- 3d model-based
- Appearance-based

3d model-based

Defines the 3D spatial decription of the human body parts. They can be classified in two large groups:

- volumetric models,
- skeletal models.

Volumetric models reproduce with high accuracy the shape of the hand or arm. Real object is often interpreted as ordered in space mesh of vertices or NURBS. This model is commonly used for computer vision purposes or for computer animation. The drawback of this approach is that is very demanding computationally and difficult to analyze in real time. With the power of today's computers, simplified models of the hand or arm (such as skeletal models) are more recommended.

As indicated earlier, instead of dealing with all the parameters of volumetric type, model can be reduced to set of equivalent joint angle parameters together with segment lengths. These are known as skeletal models. There are several advantages of using this type:

- Simplified model with the most important parameters allows the detection program to focus on the significant parts of the body.
- Due to the smaller amount of data, processing algorithms are faster.

Appearance-based

The second group of models do not use direct description of the spatial object points, because this model is based on shape of hands or arms in the visual images. The gestures are modeled by relating the appearance of any gesture to the appearance of the set of predefined, template gestures [32]. In this group distinguished a large variety of models. The most used 2d models are:

- color based model – in general, using body markers to track the motion of the body part,
- binary silhouette based model – models based on the geometric properties of the object silhouette,
- deformable gabarit based model – they are generally based on deformable active contours,
- motion-based model – based on the motion of individual pixels or image part description.

3.2.3 Gesture recognition methods

As was written earlier, one of the main issues of gesture recognition is decision problem. Currently several solutions has been proposed, which can be used regardless of device type or data representation for different classes of gestures. Classification of gestures, which should be taken into account while choosing gesture recognition method was being chosen is static and dynamic division. For each of them may be used different tools due to the different properties of these gestures. In the case of static gesture recognition, an important feature is arrangement of object, which performs gesture. In other words how the individual parts of the object are arranged in relation to each other. For dynamic gestures – as described in the previous subsection – a very important feature is the variation in time (dynamics of the gesture or dynamics of individual parts of the object performing the gesture).

To recognize static gestures, general classifier, neural network or template-matcher can be used. Methods which are capable to recognize dynamic gestures have to take into account an aspect of time. The example of this kind of method is Hidden Markov Model.

Static gesture recognition

Different techniques to perform accurate static gesture recognition have been decribed in the literature. The most common methods are neural networks, support vector machines and simple pattern techniques [43].

An neural network (NN) is an information-processing system that has been developed as generalizations of mathematical models of human cognition or neural biology. A neural network is characterized by its pattern of connections between the neurons, method of determining the weights on the connections, and its activation function [14]. They can be used both for static and dynamic gestures.

Hasan et al. [19] presented hand gesture recognition based on shape analysis. Tests were conducted for six static gestures using multi-layer perception of neural network and back-propagation learning algorithm. NN architecture consisted of one hidden layer (100 nodes), 1060 inputs and 6 output for each gesture. They achieved a recognition rate of 86.38% for a training set of 30 images and a testing set of 84 images.

Xu et al. [52] developed virtual training system of Self-Propelled Gun based on static gesture recognition and hand translations and rotations. The input data for algorithms was captured using a 18-sensor DataGlove. To recognize gestures was used feed-forward neural network with 40 nodes in single hidden layer, 18 input and 15 output nodes. The back-propagation using a variable learning rate is selected as training method. The tests were conducted on a set of 300 hand gestures from five different people – 200 gestures for training set and 100 for testing set. With the use of these methods the authors reached gesture recognition performance of 98%.

In publication of Stergiopoulou and Papamarkos publication [47] can be found static gesture recognition through other type of neural network – Self-Growing and Self-Organized Neural Gas (SGONG). To quote the authors, SGONG is innovative neural network that grows according to the morphology of hands in a very robust way. The algorithms were tested for 31 hand gestures that derive from the combination of shown and hidden fingers. Data were collected from a camera, and the recordings of hand were created in a vertical position on uniform background. With these assumptions, gesture recognition rate of 90.45% have been reached but required average computation time was about 1.5 s, using a 3 GHz CPU.

Support-Vector Machine (SVM) is a classification method invented by Vapnik [8]. SVM is a supervised learning algorithm used for classification and regression analysis, based on the mapping of characteristics extracted from instances namely the feature vectors to points in space. SVM constructs in multi-dimensional space, set of hyperplanes, which non-linearly divide points in this space (input vectors) to different classes. Support Vector Machines can be called a maximum margin classifier, because the resulting hyperplanes maximize the distance between the 'nearest' vectors of different classes. These "nearest" vectors are called support vectors.

Chen and Tseng [6] presented system based on training SVM which allows effective recognition gesture in popular game, rock-paper-scissors. One of the challenges of their work was to teach the classifier to recognize multiple-angle hand gesture. The collection of training and testing data were images from video camera, which are preprocessed using convertion to grayscale and histogram equalization. Data were collected from 5 different people for the right hand only. For the learning set consisted of 420 images and testing set of 120 images, the recognition rate of 95% was achieved.

Rahman and Afrin [38] presented hand gesture recognition system which recognizes static hand gesture for alphabet of 10 letters using Biorthogonal Wavelet Transform and SVM. Input data in the form of images – in addition to filtering – are transformed by the Canny edge detection method and then processed sequentially through Radon and Biorthogonal Wavelet Transformations. Finally, the data in this form are transmitted to the SVM classifier. To achieve robustness of the method to varying conditions, authors used a large dataset – 800 positive samples and 1500 negative image samples. Average recognition rate was 87.4%.

Liu et al. [30] proposed recognition method based on SVM and Hu moments which applied to Chinese Driver Physical Examination System. For collection of 2416 positive samples and 3050 negative samples from 20 people recognition rate of 96.5% have been reached.

Ren and Zhang [40] proposed other recognition method named by them as MEB-SVM. This method combines the SVM with minimum enclosing ball (MEB) and – according to the authors – allows to reduce computation with effective separation of all kinds of vectors in hyperspace. The input data used to test this method are images which are initially binarized and then countour

line is retrieved. Finally, contour line is converted by means of Fourier transform, so that data are independent of translation, rotating and zooming. Their method achieved a recognition rate of 92.9%.

Dominio et al. [11] presented novel hand gesture recognition based on depth data using Kinect device. The proposed processing consists following, main steps: extraction hand region from the depth map and subdivided it into palm and finger samples, extraction set of features based on finger tips and center of the hand, classification by SVM. Based on 1,000 different depth maps with 10 gestures performed by 10 different people, they achieved mean recognition rate of 99.5%.

Other popular methods are *simple pattern recognition techniques*. This group includes methods based on a simple comparison the characteristics of new problem instance with instances seen in training, instead of performing explicit generalization. In the case of gesture recognition, output information of algorithm is evaluated on the basis of similarity of the gesture to other pre-defined or learned gestures, for which belonging to groups is known. Basis on this, it is concluded that the newly read gesture belongs to the group. These techniques are generally based on a efficient lazy learning methods such as instance-based learning methods. In the context of gesture recognition, the most widely used algorithm is the k-nearest neighbour.

Ziaie et al. [57] combine naïve Bayes classifier with k-nearest neighbors algorithm for static gesture recognition purposes. Based on the dataset from the camera consisting of 580 samples for the 3 gestures, authors have achieved 93% of recognition rate.

Chang et al. [5] proposed new approach for static gesture recognition based on Zernike moments (ZMS) and pseudo-Zernike moments (PZMs), which provide greater independence from the translation of gestures. For gesture classification, k-nearest neighbor has been used, which directly processes the data from the ZMS and PZMs. In addition, authors used a minimum bounding circle, which supports the decomposition of hand silhouette into the finger parts and palm part. For dataset of 600 samples with 6 gestures performed by 10 people, gesture recognition rate of 97.3% has been reached.

Dynamic gesture recognition

Referring to the previously proposed classification of gestures may be considered that dynamic gestures include a wide range of existing types of gestures and have an important role in interpersonal communication as well as in HCl. Therefore, many approaches were proposed to gesture recognition taking into account the temporal aspect of gestures. Shen et al. [44] propose following division of these approaches:

- model-based methods,
- exemplar-based methods.

The first one includes methods that are based on the 3D model-based gesture representation, and which assume that the hand was detected and hand movements are being tracked. Model-based approaches include Hidden Markov Models (HMM), Finite State Machines (FSM), Time Delay Neural Networks (TDNN) and self-organizing networks, which preserve topology.

Exemplar-based is group of methods, which performs recognition by exemplar or template matching. These methods use a visual representation of data such as spatio-temporal local features, motion trajectiories, bag-of-features representation and other. However, this approach of dynamic gesture recognition will not be considered in thesis.

First model-based method is Hidden Markov Models. HMM is considered as specific form of dynamic Bayesian network and is doubly stochastic process with an underlying stochastic process

that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [37]. This model is represented by set of finite states connected by transitions. Each state is characterized by the state transition probabilities and probability distribution of emitting output tokens. HMM is one of the most commonly used methods for dynamic gesture recognition.

The use of HMM in the context of gesture recognition has been proposed by Yamato el al. [53]. In this approach, a discrete HMM and image feature vector sequence converted to symbolic sequences by vector quantization have been used to recognize six classes of tennis strokes.

Yang and Xu [54] proposed use of multi-dimensional Hidden Markov Model for handdrawn gestures recognition purposes. For tests, 9 gestures representing numbers from 1 to 9 was defined. Based on the 6-states Bakis model, the training set consisting of 100 samples and a test set of 450 samples, achieved recognition rate 99.78%.

In publication of Starner and Pentland [45] HMM was used for real-time recognizing sentence-level American Sign Language. The authors achieved a processing speed of 5 frames per second with recognition rate of 99.2% for 494 sentences.

In recent work also proposed new approaches and improvements to HMM method, such as using semantic network model (SNM) which introduces semantic states [39], hidden state conditional random field model [49] and non-parametric HMM for exemplar-based gesture recognition [12].

Davis and Shah [9] used FSM for hand gesture recognition using model based approach. In their approach, fingers tracking has been applied to determine the motion paths and to find the start and stop positions of gestures. Gesture was modelled as set of vectors with motion key, using motion correspondence algorithm.

Hong et al. [21] also applied FSM for gesture recognition using sequences of states in spatial-temporal space as gestures model. Each state is represented by multidimensional Gaussian.

Another method is TDNN, which is a group of neural networks with a special architecture. The main feature of this approach is that recognition is independent of the time offset of the sample. TDNN operates on sequential data.

Yang and Ahuja [56] applied TDNN to recognize American Sign Language (ASL), just like Starner and Pentland in the aforementioned publication. They provided extracting and classifying of two-dimensional dynamic gestures. Using motion trajectories, multiscale segmentation and affine transformations authors achieved 96.21% for testing set of ASL gestures.

The last approach is self-organizing networks, which are characterized by automatic adapting to the input data. This allows to dynamically adapt the network to new data.

Florez et al. [15] presented approach based on self-organizing neural network, which are capable of characterizing hand posture as well as movements. Their method reached recognition rate of 97.08% for 12 gestures.

Chapter 4

Leap Motion controller

4.1 Controller

Leap Motion is an USB sensor device, designed to provide realtime tracking of objects in three-dimensional space. It allows user to get information about objects located in devices' field of view (about 150 degree with distance not exceeding 1 meter).

Details of how Leap Motion performs 3D scene capturing, have not been revealed by Leap Motion, Inc. Three infrared LEDs are used for scene illumination, while two cameras spaced 4 centimeters apart capture images with 50-200 fps framrate, dependant whether USB 2.0 or 3.0 is used.

– photo –

4.2 Data access

Unlike Microsoft Kinect, Leap Motion does not provide access to raw data in form of a point cloud. Captured data is processed by propietary drivers supplied by vendor and accessible using an API. Leap Motion was intended to be a human-computer interface, not general purpose 3D scanner, so it is optimised for recognizing human hands or pointy objects.

The main data container we get from Leap Motion API is a Frame. Average framerate while using dual core laptop and USB 2.0 interface, is 50 frames per second. One frame consists of hands, fingers, pointables (objects directly visible by controller), and additional informations, like gestures recognized by simple built-in recognition mechanism, frame timestamp, rotation, translation and scaling data.

For purposes of this project, we have created our own data format. It contains only information necessary for us and allows us to easily save captured frames to file, and read them later for processing and testing purposes.

Chapter 5

Gesture recognition for Leap Motion

5.1 Classification of gestures

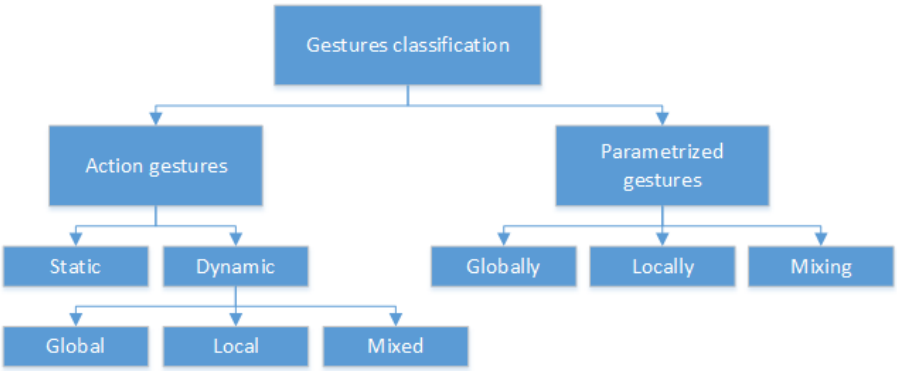

FIGURE 5.1: Classification of gestures proposed in the thesis

In section 3.1 have been presented the three main classifications of gestures based on the existing literature. Particularly important for further research and to properly define the problem of gesture recognition may be the last of the presented taxonomies, which defines a wide range of possible gestures occurring in human-computer interaction.

It should be noticed that the previously mentioned classifications are defined strictly due to the characteristics of individual gestures, which can make it difficult to analysis of choosing methods for gesture recognition. Therefore, it was decided to propose a new division that will be based on early taxonomies and in addition will be focused on the gesture recognition methods and will be defined strictly in the context of gesture recognition.

The proposed classification distinguishes basic division of gestures due to conveyed informations:

- action gestures,
- parametrized gestures.

The first group (1), represents gestures, which their recognition rely only on detection of gesture occurence. This means that action gestures have assigned some meaning, and the occurrence of this type of gesture implies execution of pre-defined action. It should also be noted that gestures of this type do not convey any additional parameters which define or describe the action. An example of this type can be gesture showing open hand, which could mean stop playing music. Another

example might be a moving index finger in circles, which means rotating previously selected object of the specified number of degrees.

Gestures, which may be included to parameterized (2), are recognized on the basis of gesture detection and returning parameters associated with the context of the gesture. In contrast to previous group, parametrized gestures carry additional meaning or parameters needed to perform the gesture. An example of this kind of gestures might be a similar gesture to the previously mentioned instance of action gestures – making circles through moving finger – but in this case – in addition to returning information about recognized gesture – a value of angle, which should be used to rotate selected object, will also be conveyed.

Then action gestures (1) were divided into static (1.1) and dynamic (1.2). The first one relates to gestures, that are not variable in time, therefore hand posture, its position and rotation are fixed during hand gesture performing. It should also be mentioned that these gestures should be independent of the orientation relative to the environment and the occurrence of this gesture is detected only on the basis of hand posture.

Dynamic action gestures (1.2) refer to group of gestures, which are continuously and non-parametrized variable in time in terms of hand posture, position and rotation in space. This group is further divided into global, local and mixed. Global were detected based on a fixed hand posture and specified movement, which may relate both to changes in position and rotation. In the case of local, hand posture is only variable in time, while the mixed include gestures, for which both the hand posture, as well as the position and rotation vary with time.

For parameterized gestures (2) distinguished division of parameterized locally (1.1), globally (1.2) and mixed. Parameterized globally include gestures, whose parameters are determined by values of position and rotation or its changes of the hand. For instance, swipe gesture can be considered as globally parametrized gesture, where the parameter is the length of the swipe motion. Recognition of globally parametrized gestures is performed on the basis of unchanging hand posture or specific changes in the shape of the hand posture over time.

Locally parameterized group (1.2) include gestures whose the hand posture or its changes are parameterized, for example a distance between index finger and thumb of one hand may be a parameter for scaling gesture. Gestures of this kind should be independent of the position and rotation in space. Recognition should be based on the hand posture, which can be problematic when it will be time-varying.

Mixed parametrized group (1.3) represents gestures which both

- hand posture or hand posture changes over time,
- and hand posture values or changes of position and rotation

are parametrized. Gestures of this group are recognized based on hand posture. Classification presented above relates to the previously described taxonomy proposed by Aigner et al. at follows:

- pointing gestures – represented by mixed parametrized gestures,
- static semaphoric – represented by static action gestures,
- dynamic and stroke semaphoric – represented by dynamic action gestures,
- static iconic – for demonstrating the shapes of objects represented by static action gestures and for presenting the size of object by locally parametrized gestures,
- dynamic iconic – mainly represented by dynamic action gestures, but can also be globally parameterized gestures for indicating the size of the objects,

- pantomimic – represented mostly by mixed dynamic action gestures, assuming that they are always performed in the same way,
- manipulation – represented by all types of parametrized gestures.

5.2 Gesture data representation

LeapSDK provides built-in classes representing real–world object seen by the controller. The basic data unit we get from Leap Motion is a Frame. Frame contains objects like Hands and Pointables (Fingers and Tools), described by features directly related real attributes.

Hand is an object representing a regular human hand. It contains Fingers, and is described by three dimensional values, like: position of center of hand, normal vector and direction vector (pointing from the center to the end of fingers).

Pointables are objects like Fingers or Tools (which are longer and thinner than Fingers). Both are described by the same set of features: position of tip, pointing direction vector, length and width.

- Frame:
  - Hands (position of center, normal vector, direction vector)
    - Pointables: (tip position, pointing direction, length, width)
      - Fingers
      - Tools

All positions are expressed in milimeters, relative to position of controller which is always located in the center of the 3D space. Manufacturer claims, that the accuracy of device is about 0.01mm. Experiments shown results better than 0.2mm, using an industrial robot moving an object in controllers' field of view. This is more than enough, as accuracy of positioning human hand is about 0.4mm. [16]

5.3 Additional processing steps

Stability of images acquired by Leap Motion can vary, which occurs as noise on captured data – sometimes fingers may flicker, or nonexisting objects appear in view. Those malfunctions happen for a very short period of time, usually less then 5 frames.

To improve data stability, a simple preprocessing has been created. Its principle is based on median filter – it uses a window with custom definied size. For a given frame, for every unique finger which have been captured in the the window, algorithm checks if it should be present in current frame, by checking its neighboor frames within defined radius r. It counts occurences of a finger (f_0) in a neighboorhood and decides:

- if $f_0 > r$, a finger should appear in that frame
- otherwise, there should not be a finger

If the finger does not exist in current frame and a check indicates it should, its position is calculated using finger positions in two closest frames where that finger appears, using linear interpolation. Otherwise, a finger is simply removed.

Using data preprocessing causes delay in data transmission to processing threads, proportional $r + 1$. For example, while capturing with 50 frames per second, with window radius $r = 4$, the delay would be:

$$\frac{r+1}{fps} = \frac{5}{50} = 0.1 second$$

Chapter 6

Static gestures recognition

As mentioned in Subsection 3.2.3, the gestures can be divided into two main groups: static gestures and dynamic gestures. The static gestures can be understood as a position and orientation of fingers and a hand in a single moment. The dynamic gestures are defined as a movement of hands in time. The problem of static gesture recognition is a subject of this chapter. The proposed approach is presented, followed by the introduction to the evaluation scheme. In the last section, the performed experiments are described, which were used to examine the effectiveness of proposed approach.

6.1 Proposed methods

The static gesture recognition problem can be stated as a problem invariant to time. That means that for each detected hand, it's position and orientation can be treated as a new data point, uncorrelated to previously shown gestures. With this assumption, one can easily generate multiple samples from the sensors in short time, but it also gives an opportunity to look at the static gesture recognition problem as a problem of classification.

In literature, most of the approaches uses two-dimensional representation of gestures, which makes recognition problems relatively simple and therefore allows to use simple classification algorithms, e.g., k-NNs [17]. The three-dimensional representation of data is more complicated to model by the set of features and finally successfully label. The 3D data yield by the sensor is represented in the sensor's coordinate system. Due to this fact, even a small change of the position or orientation of a hand with respect to the location of the sensor can negatively affect the gesture recognition system. All of those distortions should not prevent the system from recognizing those gestures if they are similar. To meet those requirement, it is needed to define what is the "small" change in orientation resulting in treating the static gestures as the same one.

In application of the library presented in this thesis, it is assumed that the learning process can be done offline, while strict online requirements has to be met for the recognition part.

To meet listed requirements, the Support Vector Machines (SVMs)[8] were used as a robust classification algorithm. The SVMs were chosen as there exist a solid mathematical background supporting the simple idea of maximizing the margin between classes. The SVMs are popular in the recent research trends and are commonly used in biology, robotics or IT for solving data classification problems. Another advantage is the existence of the open-source library libSVM [4], which contains an efficient implementation of the SVM in many programming languages. In the original work, SVMs were designed to be able to classify between two classes, but the idea was expanded to utilize the one-vs-all classification allowing to differentiate between multiple classes. This might be a drawback, as for n classes, it is necessary to train n SVMs, each to classify

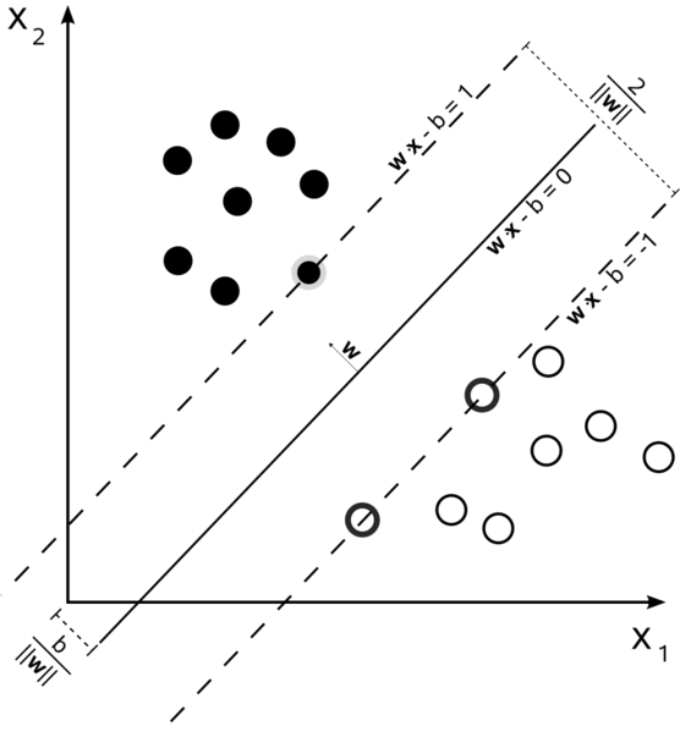

FIGURE 6.1: SVM is a classification algorithm looking for the hyperplane that maximizes the margin between classes[1]

one class against all samples from the rest of the classes. The efficiency of SVMs depends on correctly choosing the kernel function used to map the separation problem into higher dimension with expectation to achieve a classification problem that can be separated. There exists several kernel functions:

- linear: $K(x_i, x_j) = x_i^T x_j$,
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$,
- radial basis function (RBF): $K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2), \gamma > 0$,
- sigmoid: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$,

where γ, r, and d are kernel parameters. According to the authors of the library [4], linear kernels perform the best, when used for linearly separable problems, while RBF kernels are the most versatile ones and are recommended for most of the applications.

The problem of classification assumes that each sample consists of set of features, which describe a sample and can be used to distinguish it from the other samples. Additionally, each sample has a known or unknown label, which define the membership of the sample to the classification class. The samples with known labels can be used to train the classification system. The trained classification system is used to predict the labels for the unlabelled samples.

In application of gesture recognition the classification is divided into two modes: the training part (offline) and the recognition part (online). The static gesture processing flow is presented in Fig. 6.2. In the training part, the library is provided with the recorded, labelled samples of static gestures. From those samples, the sets of features are computed, which are used to train the classifier. The recognition part assumes to have trained classifier. The recognition part is provided

[1] `http://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`

with samples of static gestures without labels. For each sample the sets of features are computed and then given as input to the trained classifier. The classifier provides the information of the predicted gesture's class membership (label) for each input sample.

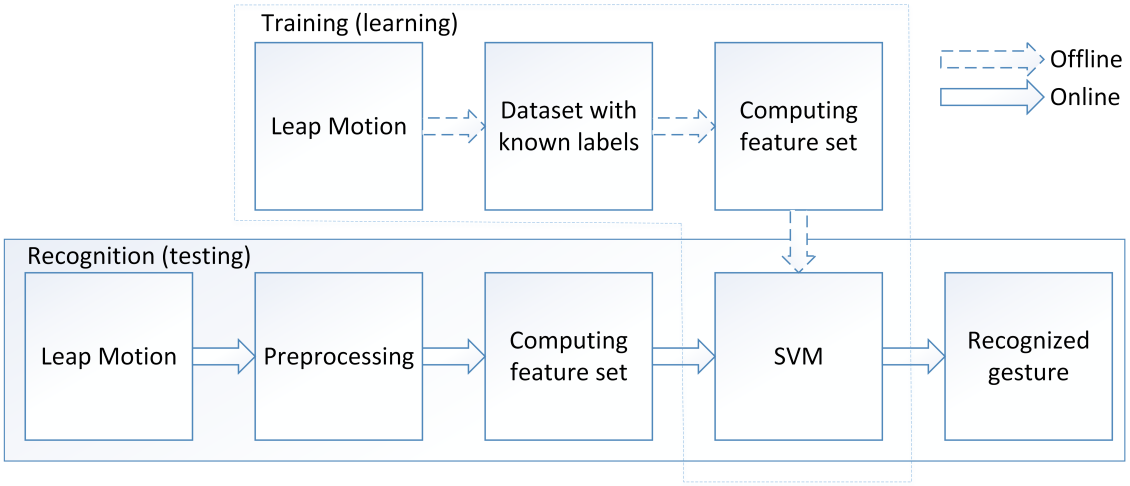

FIGURE 6.2: Proposed solution blocks for learning and recognition parts of static gestures recognition problem

While the presented approach can be treated as state-of-the-art approach it still cannot be used without defining proper feature sets for gesture recognition. The naive solution would be to use the raw data from Leap Motion sensor as the feature set. This solution was tested, but provided poor results as the raw data points are dependent on the position, orientation and scale of hand. Even a small movement in any direction resulted in decrease of classification accuracy. The literature suggests to compute a set of features invariant to wanted transformations, which can allow to fully distinguish between different classes [3, 4]. To achieve feature set with those properties, is has to been introduced taking into account the application. Also, as the gesture recognition problem using the data similar to the data provided by the Leap Motion sensor has not been yet examined by the researches, seeking right feature set is a task of the thesis. The task is approached in experimental Section 6.3.

6.2 Evaluation Methodology

6.2.1 Assumptions

To provide a user with a library working in different conditions, it was assumed that a gesture is treated as the same one independently with respect to the translation, rotation and scale of the hand. This assumption means that the static gesture rotated by unknown angles, translated in the sensor coordinate system and also with different hand sizes should still be recognized as the same gesture. Invariance to the rotation, translation and scale poses a great challenge to the recognition, but allows the future users of API to fully utilize the power of the library. It is worth mentioning that, it does not reduce the possible applications of the library, as an assignment of a static gesture to an already defined class allows to find the transformation between the model of the class and the observed gesture. This transformation can be used to examine the parameters of the instance of the model, e.g., rotation of the hand.

6.2.2 Recorded Datasets

To propose and test the quality of the features twelve static gestures were chosen:

1. the peace sign,
2. a fist,
3. full hand with space between each finger,
4. "I love you" sign from American Sign Language,
5. sign "gun" created by putting thumb and forefinger up, while holding the rest fingers in a fist,
6. all fingers in a fist with exception of thumb, which is up,
7. the sign X made with the forefingers of both hands,
8. the sing "Time" used e.g. by coaches in basketball games.
9. sign simulating rotating a knob by two fingers,
10. sign simulating rotating a knob by five fingers.

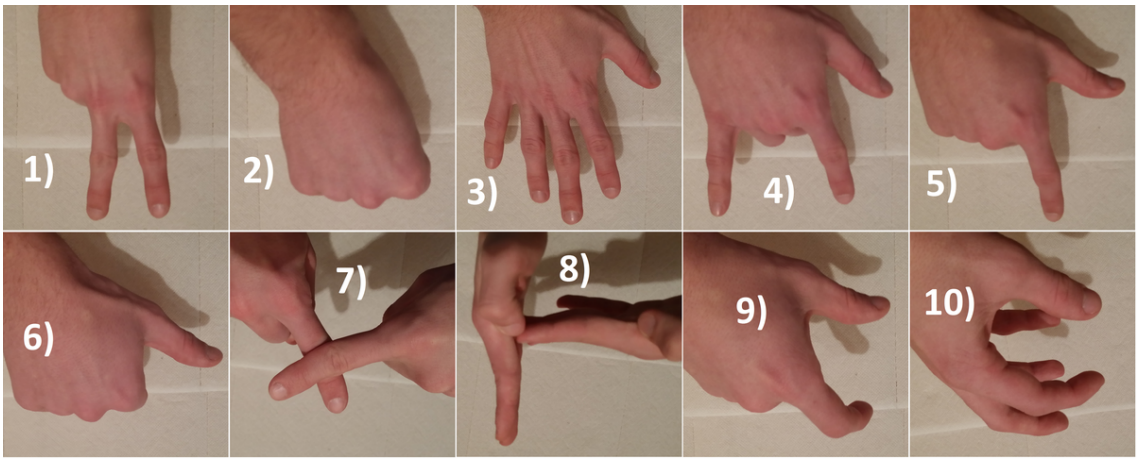

Figure 6.3: Recorded static gestures used for evaulation

The gestures are also presented in Fig. 6.3.

The sample data of each gestures were recorded using the continuous mode of recording, while moving the hands in different directions and changing the orientation of the hands. The hands were moved and rotated as each repeated static gesture is always a little bit different from the previously recorded samples. In total, for each of the proposed classes, each author of this thesis recorded approximately 1000 samples.

Having samples with known labels, the whole dataset was separated into training and testing sets in relation $2:1$. For the training, the k-fold cross-validation (CV) scheme was used, which searches for optimal C and γ parameters of SVM. This method is used to find the optimal parameters of the classification system, while estimating the performance is done on the testing data not used in the training part. In a standard version of the CV, the gathered data is divided into two sets: one containing $k-1$ parts of the data and the other 1 part of the data. The first is used to train the classification system, while the rest of the gathered data is used to estimate how well

the system is learnt. In the next iteration, the CV parts are shuffled by still having the ratio of training and validation sets. After the training, the final performance is estimated by calculating the percent of correctly recognized labels to the total size of the testing set.

6.3 Experiments

6.3.1 Evaluation of feature sets

The goal of the first experiments was to find the proper set of features. Several feature sets were proposed and tested. The first vector of features consisted of:

- number of fingers in a frame,
- euclidean distances between consecutive finger's tips,
- absolute angles between consecutive fingers.

This feature set did not take into account the relative position of fingers to the hand. The second feature set is the first feature set extended by the distances between consecutive finger tips and the position of the hand's palm. The third feature set contains features from the second feature set extended by the five angles between fingers and normal of hand's palm. The firstly tested set of all static gestures contained gestures, which were undistinguishable for the Leap Motion, because they did not take into account the way, the Leap Motion works. For gestures like fist, the recorded data contained almost no information how to classify those gestures. That's why the experiments were repeated on the five gestures, which could be easily distinguish using the data provided by Leap Motion. For this experiment the gestures peace, hand, "I love you", fist with thump up and rotating knob by 5 fingers were chosen. The results achieved by those methods are presented in Table 6.3.1.

Table 6.1: Results obtained by proposed feature sets using the libSVM library

	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
feature set 1	87.1%	87.9%	70.0%	68.4%
feature set 2	87.1%	87.9%	70.0%	68.4%
feature set 3	87.1%	87.9%	70.0%	68.4%
feature set 4	81.2%	81.0%	68.4%	68.6%
feature set 5	86.5%	85.1%	77.1%	77.5%
feature set 6	92.8%	93.1%	80.5%	81.2%

For the problem of recognition of five gestures, three first feature sets resulted in over 87% recognition rate on testing sets. The same tests for the problem of recognition of 10 gestures resulted in lower recognition rates. For feature sets 1–3 the recognition rate was below 70%, which could be unsatisfying from the perspective of the purpose of the application. The low recognition rate was analysed. The analysis revealed that the fingers are numbered by Leap Motion accordingly to the position in Z axis of the tip of the finger. This means that when finger's tips are approximately on the same position in Z axis, the numbering can change rapidly and proposed features are compared between different fingers. To achieve features that would be invariant to the numbering of the fingers, the feature set was slightly modified. Instead of containing the absolute angles and distances between consecutive fingers, it was proposed to contain the five greatest values of angles and five greatest values of distances between all combinations of finger pairings. The same sorting approach was used for the angles and distances between fingers and hand's palm. The feature sets 1, 2, 3 with sorting scheme were respectively called feature sets 4,

5, 6. Again, the same dataset with 5 and 10 gestures was used to evaluate those methods. The results are presented in Table 6.3.1. This approach was tested on the same training set and allowed to increase the recognition rate. The best results in both tasks were achieved in case of feature sets 6. The simple alleviation of finger numbering problem allowed to top the previous results with recognition rates 93.096% for 5 gesture problem and 81.235% for 10 gestures problem.

Table 6.2: Results obtained by proposed feature sets using the libLinear library

	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
feature set 1	78.3%	78.3%	50.6%	50.6%
feature set 2	78.3%	78.3%	50.6%	50.6%
feature set 3	78.3%	78.3%	50.6%	50.6%
feature set 4	78.2%	78.2%	50.7%	50.6%
feature set 5	79.5%	79.5%	55.3%	55.3%
feature set 6	88.1%	88.1%	64.7%	64.8%

While using more data and longer feature sets usually allows to achieve better results, it is worth to notice the increased learning time. In case of 5000 training samples the typical training process of radial SVM took approximately 12 hours on a desktop PC with Intel Core i7 and 6 GB of RAM. This computing time can be unacceptable by the users of the library. This is why the tests with another SVM library, libLinear [13] were performed. The libLinear's implementation of SVM utilizes linear kernels and is dedicated for large data training sets with multiple number of features. This library reduced the training time to about 5 seconds. Again, the same tests as for the libSVM were performed to compare both approaches. All achieved results are presented in Table 6.3.1. For the 5 gestures case, the libLinear achieved 88.1% on the testing set while using feature set 6 compared to 93.1% achieved by libSVM in the same condition. In this case, the libLinear might be good choice as the recognition rate is lower only by about 5%. For 10 gestures case, libLinear achieved 64.830% compared to 81.235% of LibSVM, which is a significantly lower recognition rate. It's up to user to decide which library to use, but in most recognition applications, the library can be learnt offline and only used online. That is why, the authors of this thesis recommend the SVM with RBF kernels for static gesture recognition task.

From the results obtained by the libSVM and libLinear on different feature sets, the feature set 6 was chosen as the one yielding the best results and used in further analysis.

For the problem of recognition of 10 gestures were performed an additional experiment on the test set, which presents the recognition rates achieved for each class. As a result, can be checked, for which gesturesselected feature set was the best, and for which was wrong. The fig. 6.4 shows the results of the experiment in the form of confusion matrix. Gestures of two hands fared the worst. The reason for such low recognition rate errors can be reading error from the device, which performs poorly with overlapping objects. For gesture "Time" additional problems which may affect the reading of data, can be hand position perpendicular to the Leap Motion and join the finger tips of one palm to the other. Can be also noticed that the fist is often confused with exposed thumb gesture (no. 6).

6.3.2 Preprocessing

Another factor, that might have an influence on the results is the preprocessing part, which should allow to partially remove noise from the data and thus increase the recognition rates. The preprocessing has been presented in Section **??**. The preprocessing operates in the window of hands poses recorded over time, which size can be modified. The typical library usage allowed to gather

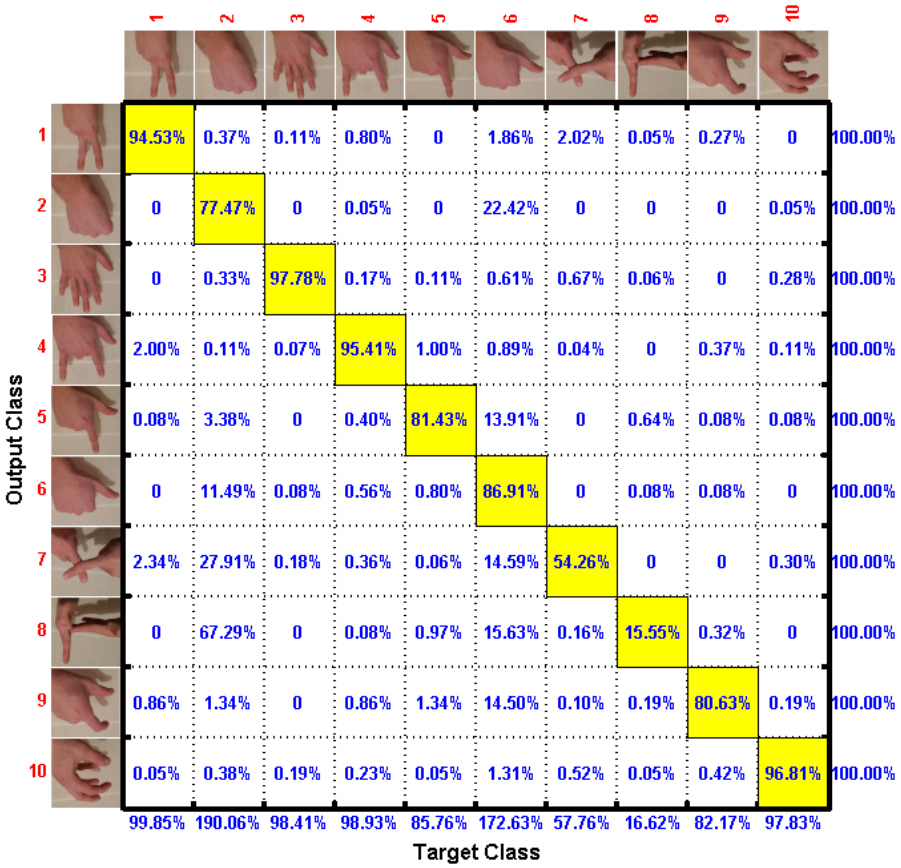

FIGURE 6.4: Recognition rates of classes for the feature set 6

data with $60Hz$, while it is assumed that the recognition can be performed with a lower framerate. The difference can be efficiently used by defining the appropriate preprocessing window. For this reason, the experiments with no preprocessing and preprocessing with width size equal to 5, 10, 15, 20, 30 were performed and the influence on the recognition rate was examined. The results are presented in Table 6.3.2. The achieved results confirm the need and importance of proper data preparation in task of data classification. For gesture recognition task containing 5 gesture poses, preprocessing allowed to increase the recognition rate from 93.1% to over 99% for window sizes equal or wider than 15. In task of recognition 10 gestures, the preprocessing allowed to increase recognition rate from 81.2% to over 84% for windows sizes equal or wider than 10. From those results, the preprocessing width of 10 was chosen as the one allowing to significantly improve recognition rate in almost any possible application of the library. The preprocessing of width 10 was used for all further experiments if not stated otherwise.

6.3.3 Postprocessing

All of already presented experiments, treated the classification results of consecutive hand poses as time-independent and not correlated with each other. In real applications, it can be safely assumed that the consecutively detected hands are similar to each other and probably define the same gesture. The remaining question to be answered was the impact of combining the consecutive recognition results for the total recognition percentage. It is important to use not only the class labels for the tested dataset, but the whole information provided by SVM containing the measure of classification rate to all possible classes. This data can be combined to measure the membership

TABLE 6.3: The recognition rate achieved with feature set 6 and different sizes of preprocessing window

preprocessing	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
off	92.8%	93.1%	80.5%	81.2%
width = 2	95.6%	96.5%	82.7%	83.2%
width = 5	95.9%	96.7%	83.1%	83.6%
width = 10	98.8%	99.0%	83.6%	84.1%
width = 15	99.0%	99.2%	84.1%	84.5%
width = 20	99.1%	99.2%	84.3%	84.8%
width = 30	99.2%	99.4%	84.8%	85.3%

rate for each class in a window of chosen width. Then the predicted class is the class with maximal measure. Formally, when there is a need to classify between k classes, the result for i-th data in dataset can be represented as:

$$l(i) = [l_{i1}, l_{i2}, ..., l_{ik}], \tag{6.1}$$

where l_{ik} is the probability measure of i-th data belonging to the k-th class. Using a window of width w, the new estimate of classes l' can be computed using the equation:

$$l'(i, w) = F(l(i - w), l(i + 1 - w), ..., l(i)), \tag{6.2}$$

where F represents the aggregation function with w vector arguments, e.g., the sum of elements of each input vector. The recognized label is the number of the vector component with the highest value:

$$\text{label} = \arg\max_k \{r(i, w)_k\}. \tag{6.3}$$

The first approach to defining the sum operator is the operator of simple adding the elements of vectors l. The second proposition is to multiple the corresponding elements of vectors l. The third approach utilizes the idea of calculating the median elements of vectors l. The three approaches were compared using feature set 6 with preprocessing width equal to 10 for 5 and 10 gestures already used in previous experiments. Simultaneously, the test were performed for different widths of window and are presented in Fig. 6.5. For almost all possible widths, the summation operator demonstrated the best recognition rate. For 5 static gestures, the summation with width equal to 10 allowed to achieve the recognition rate over 99.5% gaining over 0.5% when compared to solution without postprocessing. Interestingly, windows wider than 12 resulted in lower recognition rate than the result obtained by the window width equal to 10. For 10 static gesture recognition problem, a window of size 50 allowed to improve the recognition rate by over 5% to the value over 89%. In this case, wider window resulted in better results. Similarly to the preprocessing window size, also in postprocessing too wide window results in delayed recognition rate of a shown gesture. Also, too wide window may not lead to better results. Therefore, the postprocessing window size of 10–15 is recommended by the authors of this thesis.

Using summation as the aggregation operator treats the currently achieved result and the results from the previous as equally affecting the current recognition. Especially for wider windows, it can be assumed that the current measurement is more important than the measurement from the distant past. That is why, the idea of weighted sum was introduced. The chosen weights should have the highest weight for the current measurement and smaller values for results that were achieved earlier in time.

For this task, the weights can be taken from half of the Gaussian distribution with maximal peak reached for the currently achieved result with weights slowly decreasing for measurements

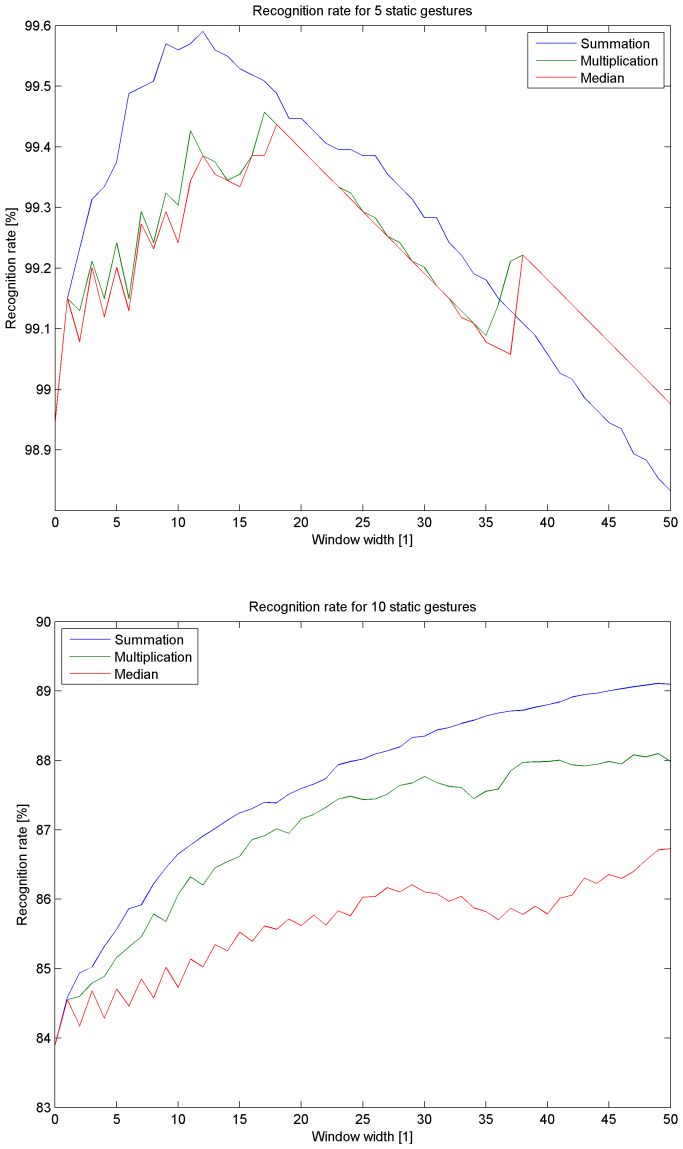

FIGURE 6.5: Evaluation of different aggregation operators for result combining in case of different window sizes

further away in time. For Gaussian, the mean was assumed to be equal to 0. The standard deviation σ was the parameter, for which different values were tested. The results were obtained on the 5 and 10 static gestures. The achieved results are presented in Fig. 6.6. For 5 gestures, small σ resulted in the weights equal to 0 for the wider window size and therefore not using the whole information available in the specified window. For greater σ, it can be observed that the achieved recognition rate is similar. The σ value equal to 10 resulted in the best recognition for almost all window sizes. For problem of 10 static gestures recognition problem, greater σ values produced results that are comparable to the results achieved with the summation as aggregation function. The usage of different weights also comes with greater computing cost. This approach did not allow to increase the recognition rate and can be omitted without effecting the recognition.

From the presented results, the summation is recommended as the aggregation operator by the authors of this thesis.

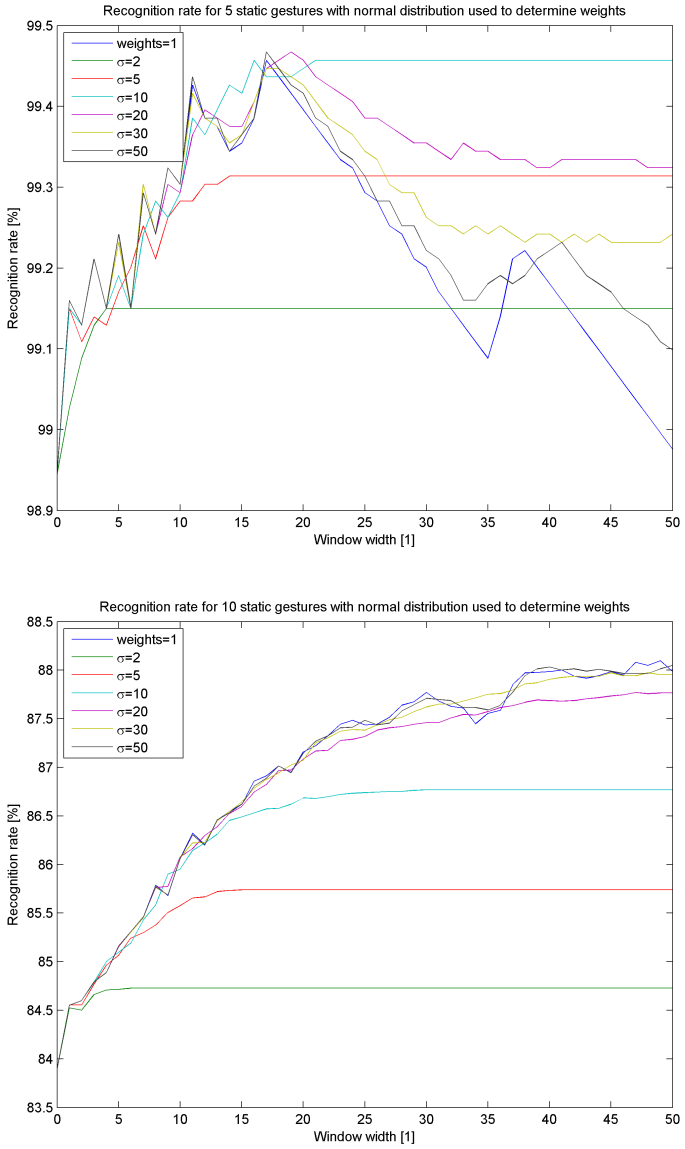

FIGURE 6.6: Evaluation of weights for postprocessing distributed accordingly to normal distribution

6.4 Finger differentiation

Finger differentiation module is responsible to verify which particular fingers are detected during gesture recognition. The module can be used for additional processing on the user side, for which the information about the visible fingers can often be significant, It can also assist interpretation of data derived from parametrized gestures. The user can also use the module to better fit dataset features to the characteristics of selected gestures.

6.4.1 Methods

For this module were used the same classification algorithms (kernel functions) as those who had the best results in static gesture recognition. As in the case of static gesture recognition libSVM library has been used and RBF kernel has been selected.

6.4.2 Evaluation methodology

In the assumptions for static gesture recognitions was mentioned that the process must be independent of the position, rotation, sizes of hand and fingers. This presumption applies also to finger differentiation. However, this assumption is still does not contain one element, from which the differentiation process should be independent – arrangement of hand. Whether forefinger is straight or bent, it is the same class, where only this one finger is taken out.

Recorded dataset

Dataset was collected by two different people. Each of them has recorded 32 classes, which reflect all the possible permutations of hand arrangements. The next step was to select from 32 gestures, those which are most frequently used by people in everyday life. Those 15 selected gestures are natural for human and do not cause a pain. People uses them as specific signes like peace sign. From the dataset were created four data collections used in the experiments:

- 32 gestures recorded by two people,
- 15 gestures recorded by two people,
- 32 gestures recorded by one person,
- 15 gestures recorded by one person.

6.4.3 Features

As mentioned previously, it is important that the fingers differentiation process is independent of things like position of hand or size of finger.

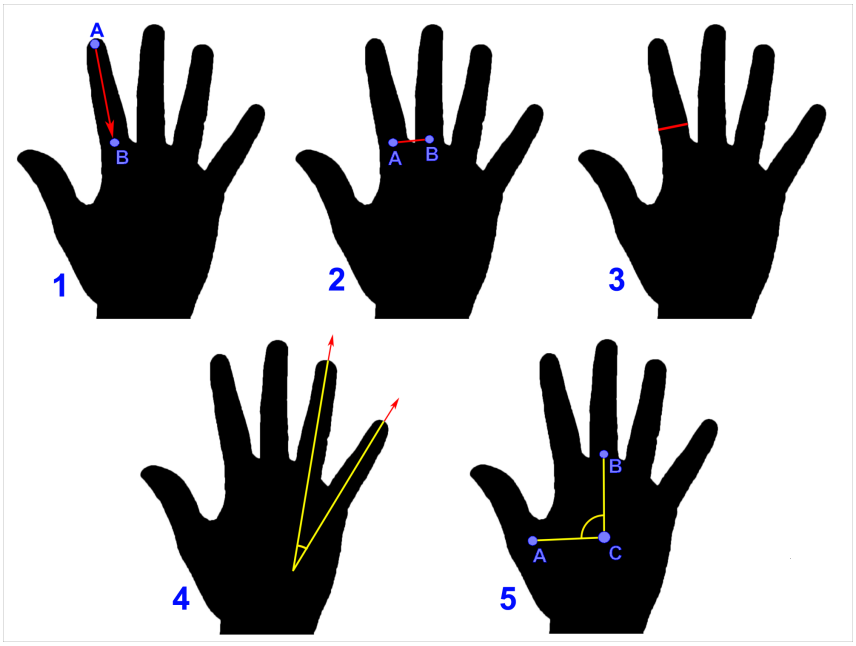

FIGURE 6.7: Features used for finger differentiation

6 features were selected, which are independent of those parameters:

1. finger count,

2. distance between two nearest base points of a finger, To appoint a finger base point has to be reversed normalized direction vector multiplied by the length of a finger. Subsequent as the beginning of the vector thus formed is set finger tip position. The end of the vector indicates the finger base point (fig. 6.7.1). An example of distance between two nearest base points of a finger is showed on fig. 6.7.2.

3. ratio of the finger thickness to the maximal finger thickness, Leap Motion controller can return information about the thickness of the fingers (fig. 6.7.3). There are known relationships between the thicknesses of the fingers, for example that the thumb is the thickest finger. Just like in the previous feature ratio was used in order to become independent of size.

4. angles between two nearest fingers, In fig. 6.7.4 this feature has been shown. It is obtained by calculating the angle between the direction vectors of two adjacent fingers.

5. angles between finger and first finger relative to palm position. To calculate this angle three positions are needed : two finger tip positions and a palm position. The next step is to determine the line segments between palm position and finger tip positions. The searched angle is between two designated segments. This angle is marked in fig. 6.7.5.

During testing new feature was created based on the features 2:
2b. ratio of distance between two nearest base points of a finger to the minimal (non-zero) distance between two nearest base points This feature is very similar to the previous one. The difference is that in this case is used ratio of distance and not the actual values. Thanks to this feature is independent of the size of hand.

6.5 Experiments

For the tests created several combinations of previously described features:

- 1,2,4
- 1,3,5
- 1,3,4,5
- 1,2,3,4,5

All sets of features has been tested for all four datasets. The percentage results can be seen in table 6.5. From the data obtained it can be concluded that the best match is when the all features are used. However, feature 2 is not independent of the size of the hand. Therefore, a new feature has been created using the ratio of two values obtained from the original feature. Then the test was performed for a set of: 1, 2b, 3, 4, 5 for all the dataset. The result for this test is in the last line of the table 6.5. Interestingly, despite the fact that the modified feature is independent of the size of hand, attribute 2b had an inferior result than the original feature, which operates on the actual value of the distance apart fingers.

It is worth to mention that it seemed that feature 4 – angles between two nearest fingers – will have a substantial contribution to finger differentiation. However, the results showed that this feature has impact only on the largest amount of data, while in other cases brought nothing.

In the second round of tests was verified how preprocessing will affect the finger differentiation process. For the best set of features from the previous test was performed a new test with preprocessing for all datasets.

TABLE 6.4: Results obtained by experimental feature sets for different data collections

features	32 classes, 2 people	15 classes, 2 people	32 classes, 1 person	15 classes, 1 person
1,2,4	75.4%	87.3%	78.4%	90.6%
1,3,5	76.3%	87.5%	79.6%	91.5%
1,3,4,5	78.1%	87.5%	79.6%	91.5%
1,2,3,4,5	83.2%	90.3%	85.8%	93.0%
1,2b,3,4,5	76.6%	87.6%	79.8%	91.6%

TABLE 6.5: The recognition rate achieved with feature set 1,2,3,4,5

features	32 classes, 2 people	15 classes, 2 people	32 classes, 1 person	15 classes, 1 person
1,2,3,4,5	%	%	%	%

Chapter 7

Detection of dynamic gestures

7.1 Proposed methods

The dynamic gesture recognition problem is a problem, where the input data consist of a list of consecutive positions and orientations of hand and fingers. Moreover, the important factor for recognition is the time dependencies between sequences of hand and finger positions. Another challenge comes with the fact, that the gestures performed slower and faster should be recognized as the same dynamic gesture.

The proposed solution utilizes parts of the solution used for recognition of the static gestures. Each frame of the captured data is described by the feature set similarly to the static solution presented in Chapter 6. The set of features for each frame is then processed by the Hidden Markov Model scheme.

7.1.1 Hidden Markov Model

A HMM can be considered a finite, N-element set of states (a graph), which are associated with the probability distribution. A Hidden Markov Model is model of a system with the Markov property. The Markov property means that the next, future state depends only on the previous state and probability of transition between those states. The first introduction of HMM comes from the work of L. E. Baum et al. [1], who proposed the mathematical background for HMMs.

The transitions between states are represented by the transition probabilities usually stored in $N \times N$ matrix T. In every state, one of the observation from the finite, K-element observation set can be generated with observation probability usually represented by the $N \times K$ emission matrix E. The finite set of all possible observation is called the alphabet. The HMM also consist of a N-element vector of initial state probabilities Π of the hidden starting state. Each HMM can be fully defined by the (T, E, Π).

The example HMM can be seen in Fig. 7.1. The HMM in the figure consists of 3 states $\{X1, X2, X3\}$ and 4 possible observations $\{y1, y2, y3, y3\}$. The probabilities a_{ij} define the transition probability from state j-th to state i-th. The probabilities b_{ij} define the observation probability of generating j observation while being in state i.

The HMM can be understood as a directed graph with two types of weighted vertices. This way, each state is represented by one type of vertices while observations can be shown as second type of vertices. There are no edges between vertices representing observations. The probability values for edges are stored in T and E matrices.

There are three main algorithms used with the HMMs:

[1] `http://en.wikipedia.org/wiki/File:HiddenMarkovModel.svg`

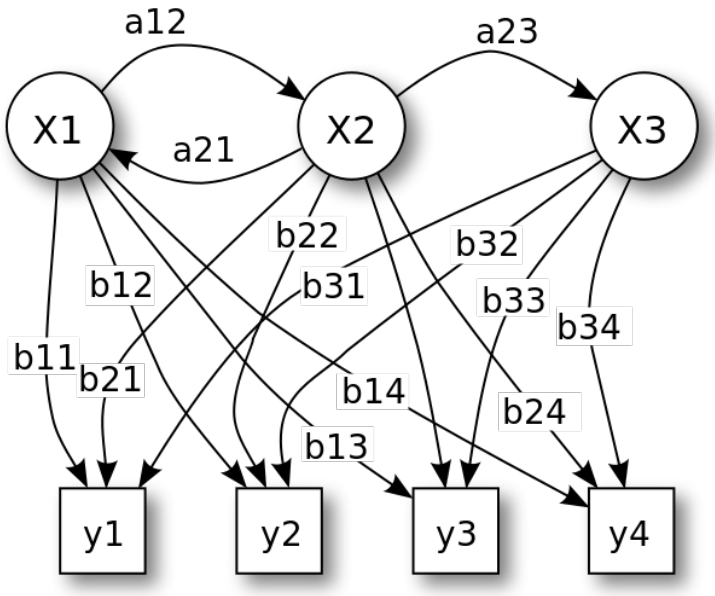

FIGURE 7.1: Solution blocks of learning and testing parts in task of dynamic gesture recognition[1]

- Forward-Backward algorithm [34, 55],
- Viterbi algorithm [34, 55],
- Baum-Welch algorithm [34, 55].

The algorithms are shortly described in the following subsections.

7.1.2 Forward-Backward Algorithm

The Forward-Backward algorithm is used to find the posterior probability of given states given the set of observations. Given the set of N hidden state variables $X = \{X_1, X_2, ..., X_N\}$ and a sequence of t observations $o_{1:t} = \{o_1, o_2, ..., o_t\}$, the algorithm computes probability of state given the observed sequence $P(X_i|o_{1:t})$. The algorithm utilizes a dynamic programming approach, by performing three steps in a loop:

1. computing forward probabilities,
2. computing backward probabilities,
3. computing smoothed values.

The forward pass phase for all states $i = 1, .., N$ computes probability $P(X_i|o_{1:k})$, where k is smaller than t, which represent the probability of ending up in state X_i after first k observations. The backward pass computes probability $(P(o_{k+1:t})|X_i)$, which are the probabilities of observing the rest of the observations from the state X_i. The smoothing part uses Bayes rule to compute the probability of state X_i given the whole observation sequence:

$$P(X_k|o_{1:t}) = P(X_k|o_{1:k}, o_{k+1:t}) \propto P(o_{k+1:t}|X_k)P(X_k|o_{1:k}). \tag{7.1}$$

The time complexity of this algorithm is $O(N^2T)$, where T is the length of observation sequence and N is the number of possible states.

7.1.3 Viterbi Algorithm

The Viterbi algorithm is used to find the most likely sequence of hidden states that best explain the set of observations. The set of those states is called the Viterbi path. The path can be found using the dynamic programming and implementing the equations:

$$\begin{cases} V_{1,k} = P(o_1|k)\Pi_k & \text{for } k = 1..N \\ V_{t,k} = P(o_t|k)\arg\max_{x \in X}(a_{x,k}V_{t-1,x}) & \text{for } k = 1..N \text{ and } t = 2..T \end{cases} \tag{7.2}$$

The solution can be understood as finding a path with maximum probability $\arg\max_{x \in X}(V_{T,x})$. The time complexity of the algorithm is $O(N^2T)$, where N is the number of states and T is the observation length and is equal to the complexity of the Forward-Backward algorithm.

7.1.4 Baum-Welch Algorithm

The Baum-Welch algorithm is the algorithm used to find the unknown parameters of the HMM. For each training example, the algorithms computes the state transition probabilities, observation probabilities in each state and initial probabilities, which maximize the likelihood of observed sequence. Having the set of sequences of observations, the algorithm can be used to train HMM to detect the sequences similar to the ones used in learning process. The algorithm uses the results obtained by the Forward-Backward algorithm to iteratively update the probabilities in the emission and the transition matrices. The training process is stopped after the desired level of a convergence is reached.

7.1.5 Structure of the HMM

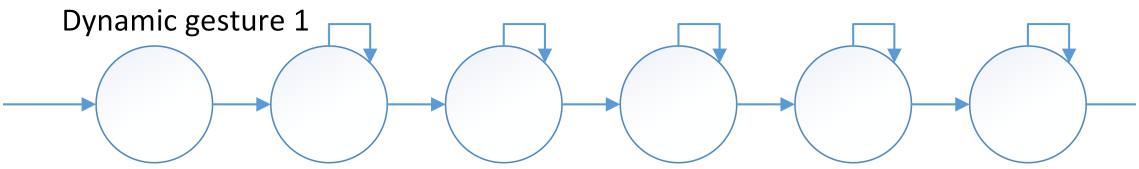

FIGURE 7.2: Structure of the HMM's states and non-zero transitions used to detect a single dynamic gesture

In the dynamic gesture recognition task, we adopted a structure of HMM where each state has non-zero transition probabilities to itself and to the next state in the sequence. The proposed structure is presented in Fig. 7.2 and was proposed by Yang et al. [55]. The states can be understood as the phases of hand movement that happen when a wanted gesture is performed. The self-transitions are used to model the different speeds of the gestures and thus allow to achieve a more robust system. This structure after training process can be used to measure the probability that the dynamic gesture occurred given the set of observations.

Having the single dynamic gesture recognition problem modelled as the sequence of N states in which k-th state is connected by the edges to the k and $(k+1)$ state and to the all observations. The problem of distinguishing M gestures translates to the problem of computing probabilities for M sequential graphs. The problem of finding whether and what dynamic gesture occurred is the

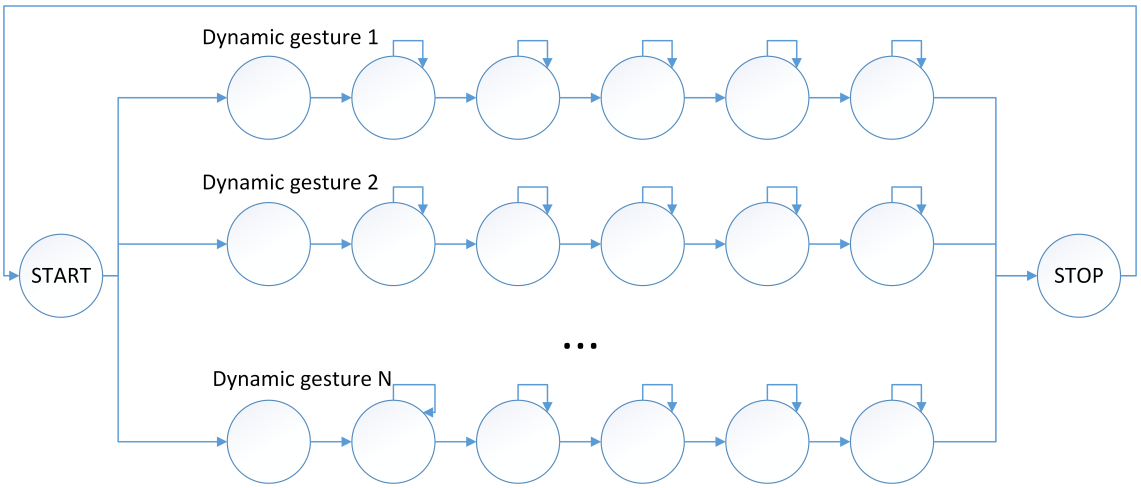

FIGURE 7.3: Structure of the HMM's states and non-zero transitions used to simultaneously detect one of M dynamic gestures

problem of finding the probabilities from each of the M HMMs. Alternatively, the M HMMs can be combined into one HMM were those single HMMs are treated as parallel paths. The structure of proposed single HMM is presented in Fig. 7.3. In this structure, the recognition process is a process of finding, which of the parallel paths is the most probable.

7.1.6 HMM observation from Leap Motion data

One of the problems when designing the system is a problem of correctly preparing the set of observations from the Leap Motion data. Most of the proposed solutions in the literature assume that the observations are one-dimensional [34, 55]. This poses a challenge as the raw data from Leap Motion is high-dimensional. This is the reason, why it is needed to reduce the dimensionality of the observation of the single hand in the predefined moment in time.

To do this, unsupervised clustering algorithms were introduced. For this task we examined 3 methods:

- Chinese whispers [2, 7],
- Girvan-Newman clustering [18],
- k-means with additional methods to determine the number of clusters. [46, 31].

Chinese whispers algorithm is an efficient randomized graph-clustering algorithm, which is time-linear in the number of edges. The algorithm was proposed in work by Biemann [2] and thoroughly examined in Natural Language Processing problems. The main advantage of the algorithm is the ability to independently determine the number of classes. In tests, we used the implementation available in dLib-ml library [28], which provides multiple implementations of machine learning algorithms.

Girvan-Newman clustering is a hierarchical algorithm used to detect clusters in a data represented as a graph. The algorithm progressively removes edges from the original network until it reaches the maximal number of iterations or an error condition is met. An adjacency matrix is used to represent the edges between data, where edge value is the similarity between two data points. The algorithm iteratively calculates the eigenvalues of a created matrix using the power iteration method. In case of a complete graph, this algorithm is relatively slow.

Another proposed approach is k-means clustering algorithm. The idea for the algorithm comes from polish mathematician Hugo Steinhaus, but the first practical application has been presented by MacQueen [31]. The algorithm initially randomly selects k starting points and iteratively performs two steps utilizing the idea of expected-maximization [10]:

1. For each data sample the algorithm calculates the distances to k centroids and labels the samples with the label of the centroid with the smallest distance.

2. For each class, the algorithm recalculates centroids position to the averaged position of all samples belonging to class.

The algorithm stops after the maximum number of iterations or when the change between two consecutive iterations is smaller then defined value. Unfortunately, the algorithm requires prior knowledge of the number of expected classes. Although there exists a heuristics methods aiding in correctly choosing the number of classes. One of those methods is plotting the error sum of squares (SSE) within classes against the number of chosen classes. In this plot the drastic drop of SSE is sough and the number of classes it happens is considered a good choice. Another, more formal approach introduces the measure of dissimilarity and is called Silhouette width [42]. Declaring $a(i)$ as the average dissimilarity with all the other data within the same cluster and $b(i)$ as the lowest average dissimilarity to any other cluster which i is not a part of, we can write a measure:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{7.3}$$

The big value of $s(i)$ implies good clustering, while small value of $s(i)$ suggests that the number of classes was not properly chosen.

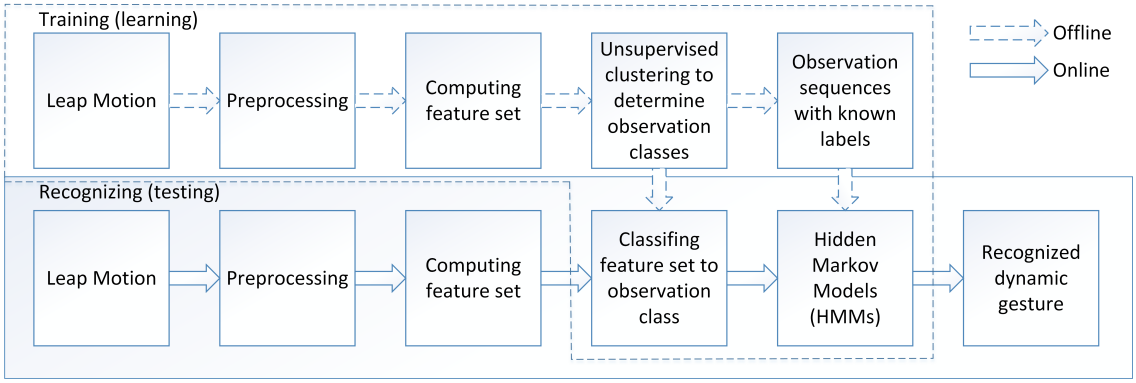

FIGURE 7.4: Solution blocks of learning and testing parts in task of dynamic gesture recognition

Finally, the whole processing flow has been designed and is presented at fig. 7.4. The whole solution consists of two parts: off-line learning and on-line recognition. For learning, the raw data is preprocessed and then the feature are extracted. Similarly to the static gestures, the features set were computed. In training part, those features are extracted from all recorded positions for all dynamic gestures. Then one of the unsupervised clustering algorithms is used to find the classes of observations. Knowledge of the classes is used to represent each dynamic gesture as a series of one-dimensional observations. After that for each dynamic gesture, the corresponding HMM is learned by running the Baum-Welch algorithm on the sequence of observations. The Baum-Welch algorithm finds the matrices that maximizes the likelihood for one sequence of observations, therefore the problem of training on multiple training data was encountered. Following by the

idea presented in [34], each matrix trained by one training example is incorporated into trained matrix by using simple addition with learning rate μ. Denoting the transition matrix by T and learnt transition matrix T_{new}, it can be written:

$$T = (1 - \mu) \times T + \mu \times T_{new} \tag{7.4}$$

The same idea applies also to the emission matrix E and learnt emission matrix E_{new}:

$$E = (1 - \mu) \times E + \mu \times E_{new} \tag{7.5}$$

The problem with the matrices by computing equations**??** is the fact, that they do not represent the probability as the total values do not add to 1.0. Therefore, each row of new matrices is normalized to 1.0. When dealing with multiple training examples, the resulting solution may be overfitted to the training samples. To deal with this problem, the K-fold cross validation was utilized in a similar approach as in the static gestures. To complete the training process, each of the HMMs is put into one recognition structure. During the recognition part, observed sequence is treated as input to the Viterbi algorithm performed on each of the HMMs and the output is analysed. The HMM with the highest likelihood is chosen, and the observed sequence is labelled with the label of this HMM.

In the on-line working, the raw data from leap motion is preprocessed, then each frame is labelled accordingly to the classes learnt by the unsupervised clustering algorithm. The next step include providing the set of observations to HMM and running the Viterbi algorithm. The Viterbi algorithm finds the most likely sequence in one the parallel paths, which informs us about the number of the found gesture. If the likelihood is above threshold, the gesture is assumed to be correctly recognized.

7.2 Evaluation methodology

To evaluate the quality of recognition achieved by proposed processing pipeline, 6 dynamic gesture were chosen for which the data was recorded. For each of those gestures, we recorded 120 samples (30 samples per each author). The recorded data were also recorded in different positions with respect to the Leap Motion coordinate system and with different speed to measure the wanted invariance and robustness. The chosen dynamic gestures are:

1. “123” – it’s a gesture when performing counting to 3 using hands,
2. “door” – it’s a gesture performed while trying to grasp a handle of door and then open the door,
3. “circle” – making a circle in the air with the forefinger,
4. “scissors” – simulating the cutting with the scissors made by a forefinger and a middle finger,
5. “gun” – only the thumb and the index finger are not in the fist. The hand moves up simulating the movement during the firing from the gun.
6. “moving the object” – performing the task of grasping an invisible object, moving it and letting it go in different place.

To test the proposed approach, similarly to the static gesture recognition problem, the $120 \times 6 = 720$ samples were divided into training set containing $66,7\%$ of recordings and testing group containing the remaining 240 recordings. The part of the gestures recorded by Katarzyna were

used to find the proper number of clusters than could be used by form the observations for the HMMs. The training set was used to train each HMM separately on the training data of each gesture. Preliminary results revealed that initialization of HMM matrices has an impact on the achieved results. Due to no prior knowledge, the random initialization was chosen. This approach did not yield reliable solutions in all situation and therefore each training process is performed 10 times and the best model from cross-validation is returned. Each training cycle of all HMMs takes between $1 - 10$ minutes, with total training part taking around $25 - 45$ minutes. If not stated otherwise, the learning rate μ was set to 0.1, k-cross validation was performed for $k = 5$ and the number of states in one HMM was set to 10.

7.3 Experiments

The performed experiments started with testing the unsupervised clustering methods to determine the correct number of observations. Based on the successful static gesture recognition, each of the hand poses was represented by the vector of values containing:

- number of detected fingers,
- 5 greatest angles between the finger tip vector and palm normal,
- 5 greatest angles between the fingers tip vectors,
- 5 greatest distances between the tip positions of fingers.

In order to compare different hand poses, the distance function was introduced as the L2 norm between feature vectors:

$$d(x, y) = \sqrt{\sum_{i=1}^{16} (x_i - y_i)^2} \tag{7.6}$$

The Chinese whispers and Girven-Newman uses the similarity function, which was defined as:

$$s(x, y) = \frac{1.0}{\max\{d(x, y), eps\}} \tag{7.7}$$

where *eps* was the numerical epsilon.

For the Chinese Whispers and Girven-Newman clustering the typical parameters from dlib-ml library were used — the Chinese Whispers maximal iterations were set to 100, while the Girven-Newman was run with maximal iterations equal to 2000 and precision eps was set to $1e - 4$. For the SSE analysis for different cluster number, the publicly available script was used[33]. The script automatically calculates the k-means clustering algorithm for chosen range of k values and presents the figures, which can be a help in determining the correct number of classes. For the Silhouette width analysis own, R script was written.

The SSE analysis usually does not yield the direct answer. The tool provides several plots, but the most interesting one is presented at fig. 7.5. Looking at this plot, there is a significant drop in SSE for the cluster number equal to 5 and 10. It is expected that either of those numbers is the correct choice when it comes to the number of classes for k-means algorithm. The Silhouette width approach estimated that the there are 4 distinguishable classes in recorded gestures.

The proposed values of clusters for different methods are presented in the table 7.1. The Girvan-Newman algorithm did not provide any estimate within 24 hours and therefore was considered impractical. Due to the inconclusive results, the correct number of classes was decided to be determined by performing experiments with different class number parameter. The analysis of the

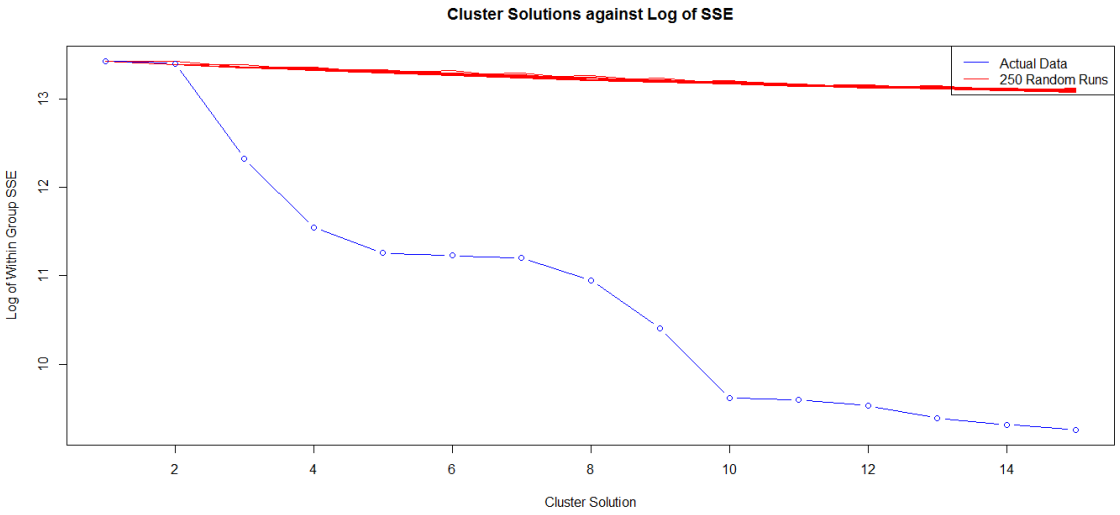

FIGURE 7.5: Different representations of SSE as a function of number of clusters for the k-means algorithm

TABLE 7.1: Comparison of suggested number of clusters for the dataset containing all positions of hand in all dynamic gesture recordings made by Katarzyna

	Girven-Newman	Chinese whispers	SSE vs clusters	Silhouette width
recordings by Katarzyna	-	613	5 or 10	4

correct class number was postponed until the correct feature set is chosen. Until stated otherwise, the class number equal to 10 was chosen.

The next challenge was to find the feature set describing one gesture in time, that would contain the information about the dynamic nature of the gesture to be recognized. The first feature set consisted of features encoding the information about the speed of the hand. For each $i-th$ hand position in recorded sequence, the $(i-10)$-th position was also considered. The feature set consisted of the recorded displacement of hand in X, Y, Z directions. The number of fingers in both positions were also added, resulting in feature set containing:

- finger count in i-th hand position,
- finger count in $i-10$-th hand position,
- displacement of hand position in X,
- displacement of hand position in Y,
- displacement of hand position in Z.

The presented approach allowed to achieve 64.583% on cross-validation set. Due to the small testing set, the total recognition rate was calculated on training and testing sets. This approach resulted in 61.806% on all samples, which is too small to by used in real applications.

The previous approach suffered from the fact, that the encoded speeds are represented in the coordinate system of the Leap Motion. Executing the gesture with different angle to the Leap Motion's coordinate system makes the recognition tasks harder and sometimes results in mislabelling. Therefore, the feature set was supposed to encode the information with respect to the local coordinate system of hand. In new approach, the magnitude of the displacement is calculated. It is independent of the chosen orientation of the coordinate system. To represent the

direction of movement, the normalized displacement vector is computed as a vector connecting the center of $i - 10$-th hand position with i-th hand position. Then, the dot product between normalized displacement vector and palm normal in $i - 10$ position is calculated. As this not encode the whole information, the dot product between the normalized displacement vector and the plan direction is also computed. The resulting feature contains also 5 elements:

- finger count in i-th hand position,
- finger count in $i - 10$-th hand position,
- magnitude of the hand's displacement,
- dot product of the normalized displacement vector and the palm's normal in $i-10$-th position,
- dot product of the normalized displacement vector and the palm's direction in $i - 10$-th position.

This approach allowed to improve the recognition rate in cross-validation (69.583%) and on the whole set (67.500%), although was still not satisfying from the practical point of view.

The simple observation on the feature set and recorded gestures reveals that sometimes only the fingers are changing positions, while position of hand is steady. The previous approaches, did not encode this information in feature vectors, which made the those changes invisible to the further processing pipeline. Therefore, the new feature vector contained also the information about the displacement of each corresponding finger. In order to keep the feature size small, it was decided to only incorporate the magnitude of those displacements. Due to the nature of Leap Motion's erratic numbering of fingers, those magnitudes were also sorted. The total of 4 top values were added to feature vector resulting in:

- finger count in i-th hand position,
- finger count in $i - 10$-th hand position,
- magnitude of the hand's displacement,
- dot product of the normalized displacement vector and the palm's normal in $i-10$-th position,
- dot product of the normalized displacement vector and the palm's direction in $i - 10$-th position,
- 4 greatest magnitudes of finger displacements.

The results were not satisfying as were even worse than the ones achieved for the feature set 2 – 66.111% compared to the previously received 67.500%. The further inspection revealed that, while adding displacement of fingers helps in most situations, it can also pose a great risk. In case of different order of finger numbering, the resulting displacement may be calculated between different fingers, thus resulting in big artificial displacement of fingers. Therefore, the idea of using displacement of fingers was abandoned.

Looking at the recorded gestures, it seemed that speed information is not sufficient to correctly detect the sequences of gestures combining in one dynamic gesture. The new feature set, contained also the information about the static hand positions allowing to distinguish dynamic gestures that are comprised of sequential static gestures. The new feature set contained the already proposed speed part with additional static encoding that was successfully used in static gesture recognition:

- finger count in i-th hand position,

- finger count in $i-10$-th hand position,
- magnitude of the hand's displacement,
- dot product of the normalized displacement vector and the palm's normal in $i-10$-th position,
- dot product of the normalized displacement vector and the palm's direction in $i-10$-th position,
- 4 greatest euclidean distances between all combination of finger's tips in i-th position,
- 4 greatest absolute angles between all combination of finger's vectors in i-th position,
- 4 greatest euclidean distances between finger's tips and palm's position in i-th position,
- 4 greatest absolute angles between finger's vectors and palm's normal in i-th position.

The experiments showed that the proposed approach is not as successful as it was believed. It was believed that the sophisticated static part of feature set is dominating dynamic part of feature set, which prevents the method from achieving better results. Therefore, the 5-th proposed feature set consist of static part represented by the 4 greatest euclidean distances between all combination of finger's tips in i-th position and 4 greatest absolute angles between all combination of finger's vectors in i-th position. Similarly, feature sets 6, 7, 8 were also the propositions containing the reduced static parts, but did not improve significantly over already received results. The results obtained by all proposed feature sets are presented in tab. 7.3.

Table 7.2: Results obtained by the HMM approach using different feature sets, 10 classes

	5 gestures, CV	5 gestures, whole dataset
feature set 1	64.583%	61.806%
feature set 2	69.583%	67.500%
feature set 3	68.542%	66.111%
feature set 4	77.917%	76.111%
feature set 5	66.667%	63.889%
feature set 6	72.912%	70.417%
feature set 7	71.875%	69.167%
feature set 8	78.125%	77.222%

The best results were obtained for feature set 8, but the results obtained by the full static representation (feature set 4) are not significantly worse. For further assessment, the feature set 4 was chosen as the one that describes each gesture with more information and therefore is believed to work better for gestures that were not included in proposed experiments.

The next experiments were performed to test what is the best number of observation classes for k-means algorithm. The tests were conducted using feature set 4 and learning rate $\mu = 0.05$.

Table 7.3: Results obtained by the HMM approach using feature set 4 and different numbers of class clusters

	5 gestures, CV	5 gestures, whole dataset
$k = 4$	77.292%	74.722%
$k = 5$	79.583%	77.639%
$k = 6$	79.583%	77.639%
$k = 8$	77.708%	75.977%
$k = 10$	77.917%	76.111%
$k = 12$	78.542%	77.361%

The best obtained results are for the $k = 5$ and $k = 6$ allowing to increase of the recognition rate from cross-validation to 79.583% and increase to the 77.639% on the whole dataset.

The next experiments involved finding the best learning rate. In our application, the stable learning rate was chosen to minimize the number of parameters to tune. The further developments may involve using learning rate that is changing depending on the already received learning rate. Using predefined learning rate may be dangerous as small value may mean small convergence, while too big step may result in not converging to the locally best solution. The achieved results are presented in 7.3. In proposed application, the learning rate $\mu = 0.05$ allowed to achieve the best results.

TABLE 7.4: Results obtained by the HMM approach using feature set 4, 5 observation classes and different learning rates

	5 gestures, CV	5 gestures, whole dataset
$\mu = 0.01$	77.500%	76.944%
$\mu = 0.05$	80.417%	79.028%
$\mu = 0.1$	79.583%	77.639%
$\mu = 0.2$	69.583%	69.7222%

The only parameter not tested in the already described experiments is the number of state each HMM is composed of. The previously used value was equal to 10 states. The results from experiments are presented in tab. 7.3. The performed experiments confirmed the choice as smaller number of states (5) resulted in lower recognition rate, while greater number of states did not improve the result. The drawback of taking HMMs with big state numbers is the learning time. The performed experiments and theoretical computational complexity confirm that the computational complexity grows linear with the chosen number of states, which in practical applications means linear grow of taken processing time.

TABLE 7.5: Results obtained by the HMM approach using feature set 4, 6 observation classes and different state elements

	5 gestures, CV	5 gestures, whole dataset
$K = 5$	73.958%	73.611%
$K = 10$	79.583%	77.639%
$K = 20$	77.500%	76.528%
$K = 30$	77.917%	76.25%

Chapter 8

LMGesture library dedicated for Leap Motion controller

8.1 Architecture

One of the main goals of creating a library for gesture recognition using Leap Motion, was simplyfying the task for end-user and creating consistent, easy to use inferface for applications use. A library can be treated like a black box which takes data frames as a input and responds only if a gesture from defined set has been recognized. Code was written using C++. Library is distributed as open source code files.

The library is divided into three main parts:

- process
- static gesture processing thread
- dynamic gesture processing thread

Process part is a middle layer between user application, Leap Motion and data processing threads. That approach improves clarity of code and makes modification easy.

Workflow:

1. User creates LeapProcess and LeapListener object.
2. User starts processing.
3. Library starts threads for static and dynamic processing.
4. Leap Motion reads a frame.
5. A frame is preprocessed and sent to processing threads.
6. If processing finds a gesture, an appropriate user–supplied method is invoked.

Elements of library:

- GestureFinger, GestureHand, GestureFrame, Vertex - classes representing respectively: Frame, Hand and Finger, with additional Vertex class, which simpifies operations on 3D points
- StorageDriver – class supporting reading data from .lmr files
- FileListener – listener grabbing input data from file and feeding them to LeapProcess

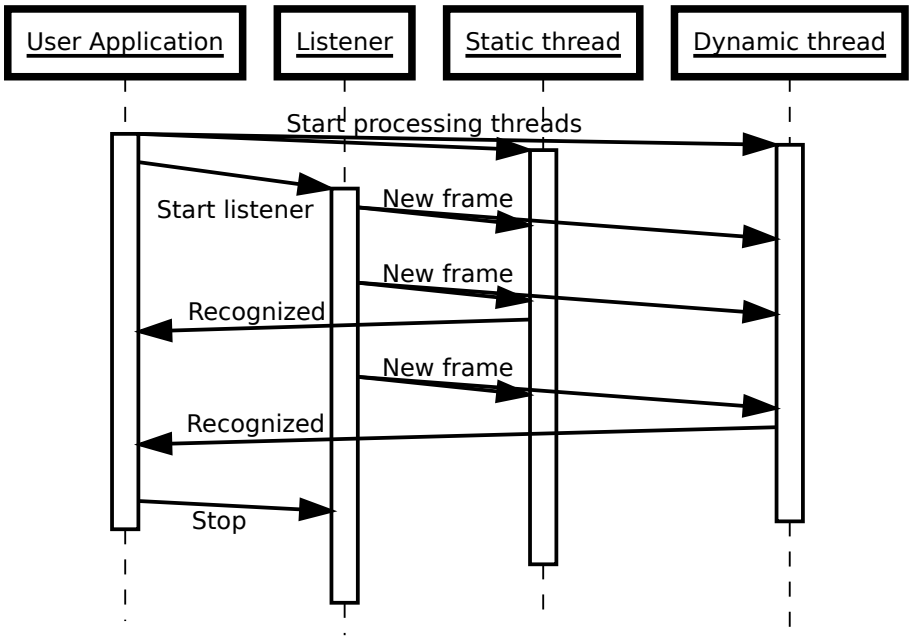

FIGURE 8.1: Sequence diagram of processing

- LeapListener – listener grabbing data from Leap Motion controller
- LeapProcess – class responsible for creating (in separate threads)
- LibLeap – main file intented to being included by user to their project
- LMpre – data preprocessing
- LMRecorder – class responsible for translating Leap Motion API data structures to proposed by us
- RecognizedGesture – abstract class, user should override its methods in order to handle actions which will execute after a gesture was successfully recognized

8.1.1 Model

While data obtained from Leap Motion Controller are being processed, they are stored using a specially created classes representing the data. The most important class is GestureFrame, which represents a single frame captured from device. All gathered data is stored in a vector containing elements of GestureFrame type. GestureFrame holds the following information:

- timestamp,
- list of data of detected hands in the frame, stored in a vector containing elements of GestureHand type.

GestureHand stores parameters of hand performing a gesture. In one instance of GestureFrame many instances of GestureHand can be stored. GestureHand holds the following information:

- hand id,
- plam position,

- stabilized palm position,
- palm normal vector,
- palm direction,
- list of fingers of particular hand, stored in a vector containing elements GestureFinger type,
- ordered value, obtained during hand sorting.

GestureFinger stores parameters of one finger. In one instance of GestureHand many instances of GestureFinger can be stored. GestureFinger contains:

- finger id,
- tip position,
- stabilized tip position,
- finger direction,
- finger length,
- finger width,
- ordered value, obtained during finger sorting.

8.2 Processes

8.2.1 The learning process

The learning process is a process of teaching LeapGesture library a new gesture by API user. First gestures must be recorded gestures in LMR format. To obtain recordings in this format user may use Gesture Recorder – a module included in the library, which records data from Leap Motion Controller and saves it as LMR file. When all desired gestures are prepared, the user starts the process of learning by [tutaj dodać informacje jak uruchomić ten prces - prawdopodobnie jakas komenda + jakies parametry typu -d uczenie gestów dynamicznych, -s uczenie gestów statycznych + pliki LMR jako argumenty].

The first step in the learning process, which is performed by the system is the preprocessing of data. Training data should be deprived of noise or should not have lost fingers for several frames. According to user-specified parameters, the system will learn the gesture as a static or dynamic gesture. System uses support vector machine (SVM) for learning of static gesture. For dynamic gestures it uses hidden Markov model (HMM). The learning process for static gestures:

1. In the first stage, data which will be used during the learning are preprocessed.
2. Data are scaled using internal scaling module. The result of this step are scale and range files. In the scale file scaled data is stored. The range file contains information that enables to scale any data in the same way.
3. Scale file with scaled data is transmitted to a training module. Data is processed using SVM algorithm. Additionally during the learning process k-validation is performed and results are returned to the user. In the process model file is created, which is used during static gesture recognition process.

[opisać learning process dla dynamicznych]

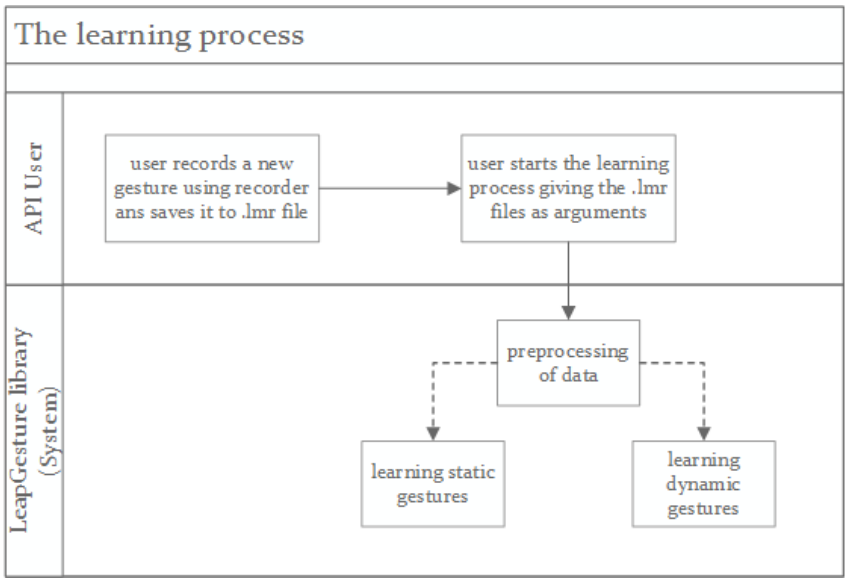

FIGURE 8.2: Diagram showing the learning process for LeapGesture library

8.2.2 The recognition process

The recognition process is a process of recognizing the gesture performed by the API user. During this process, the system uses the information obtained from the learning process. [Opis procesu recognition process, przedstawienie scenariusza uzycia w stosunku do architektury]

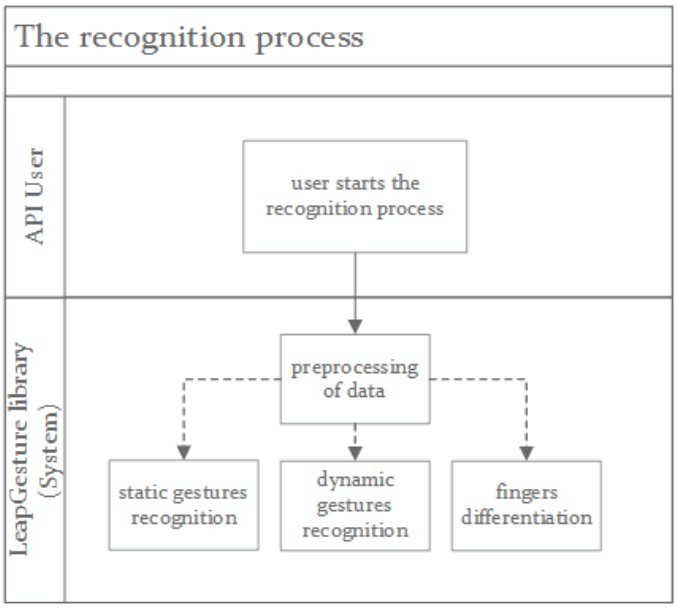

FIGURE 8.3: Diagram showing the recognition process for LeapGesture library

Modules handlers are implemented using the Observer pattern. For each of the modules are specified relevant events that are reported in the key moments of recognition such as: moment in which particular gesture began to be recognized, moment in which the particular gesture stopped being recognized, moment in which frame processing is finished. The user can handle events received in specific module by implementing appropriate listener interface and adding reference to listener list in specific library module. During gesture recognition process user can use from 1 to 3 listeners. Each listener is on a separate thread.

Methods for a static gesture observer:

- onStart – signals the start of the gesture,
- onFrame – returns a list of recognized gestures with matching probabilities,
- onGesture – signals the end of the gesture and returns a list of recognized gestures with matching probabilities.

Methods for a dynamic gesture observer:

- onStart, onFrame – have the same purposes as in the case of static observer,
- onGesture – signals the end of the gesture and returns a list of recognized gestures with matching probabilities and parameter values.

Methods for a finger differentiation observer:

- onFrame – returns a list of matched classes with probabilities,
- onChange – signals a change of fingers arrangement.

As in the learning process for recognizing static gestures system uses SVM and for dynamic gestures – HMM. For finger differentation system uses support vector machine. However, there are other classes than those used for static gestures recogniton.

8.3 Gesture recorder and visualizer

Recorder and vizualizer module is an additional part of library that allows users for easy management of gestures recordings. The recorder collects data from Leap Motion Controller, converts it into data representation described in Section 8.1.1 and writes it to the LMR file, which is supported by the LMGesture library. Visualizer enables users to see the recorded gestures stored in LMR format. Data gathered from Leap Motion, used to recognize gestures are very large and the data processed in the library have a specific format. Therefore, it was necessary to create an auxiliary program, that it would facilitate the work with data in the fastest and most user-friendly way.

8.3.1 LMR files

This is a file format specially developed for the LMGestue library, supported by various modules for example by the visualizer. The file structure is as follows:

- Line represents one frame.
- One frame contains: timestamp and hand parameters.
- Hand parameters include: hand id, palm position, stabilized palm position, palm normal vector, palm direction vector and detected fingers parameters.
- Finger parameters include: finger id, finger tip position, stabilized tip position, finger direction vector, finger length and finger width.

Example lines from LMR file:

4.78984#7 -9.15495;113.759;46.292 0;0;0 -0.157914;-0.981575;0.107585 -0.0196983;-0.105799;-0.994192f1 -64.0011;122.41;-28.5433 -64.831;125.304;-38.845 -0.574857;0.0254502;-0.817858 50.4269 13.0854

16.6761#7 -8.94345;113.525;46.2299 0;0;0 -0.158215;-0.981508;0.107745 -0.0193809;-0.106012;-0.994176f1 -63.8514;122.144;-28.6825 -64.8135;125.248;-38.6638 -0.575861;0.0261358;-0.81713 50.4225 13.0803

27.4361#7 -8.72478;113.293;46.1443 0;0;0 -0.159539;-0.981183;0.108754 -0.0203256;-0.106877;-0.994065f1 -63.5346;121.826;-28.7338 -64.7892;125.183;-38.475 -0.576776;0.0265335;-0.816472 50.3783 13.1248

This example shows a frame containing a hand with one finger exposed. The color red is used to mark frame timestamp, the blue color highlights information about hand, and the color green indicates information about the exposed finger.

Technical information regards LMR files:

- Timestamp and hands are separated by "#".
- In specified hand occurs hand parameters and fingers.
- Hand parameters are separated by space, and finger are separated by "f" (hands parameters and fingers are separated by "f" too).
- Specified finger has parameters, which are splited by " ".
- Values in the trivalent parameters are separated by a semicolon.

8.3.2 Visualizer

Visualizer presents the contents of the LMR files. As mentioned earlier, each line of the file is a separately converted frame obtained from Leap Motion Controller. In one moment only one frame is displayed. Almost all fields of the model are visualized:

- palm position,
- palm normal vector,
- palm direction,
- tip position,
- finger direction,
- finger length.

An example of parameter, which is not visualized, is finger width, in order not to obscure the image.

List of Visualizer features:

- Program can read the LMR files chosen by the user.
- The user can load many files at once and select one of the recordings from the drop-down list.
- To facilitate the user work with visualizer, program has implemented windowing interface.

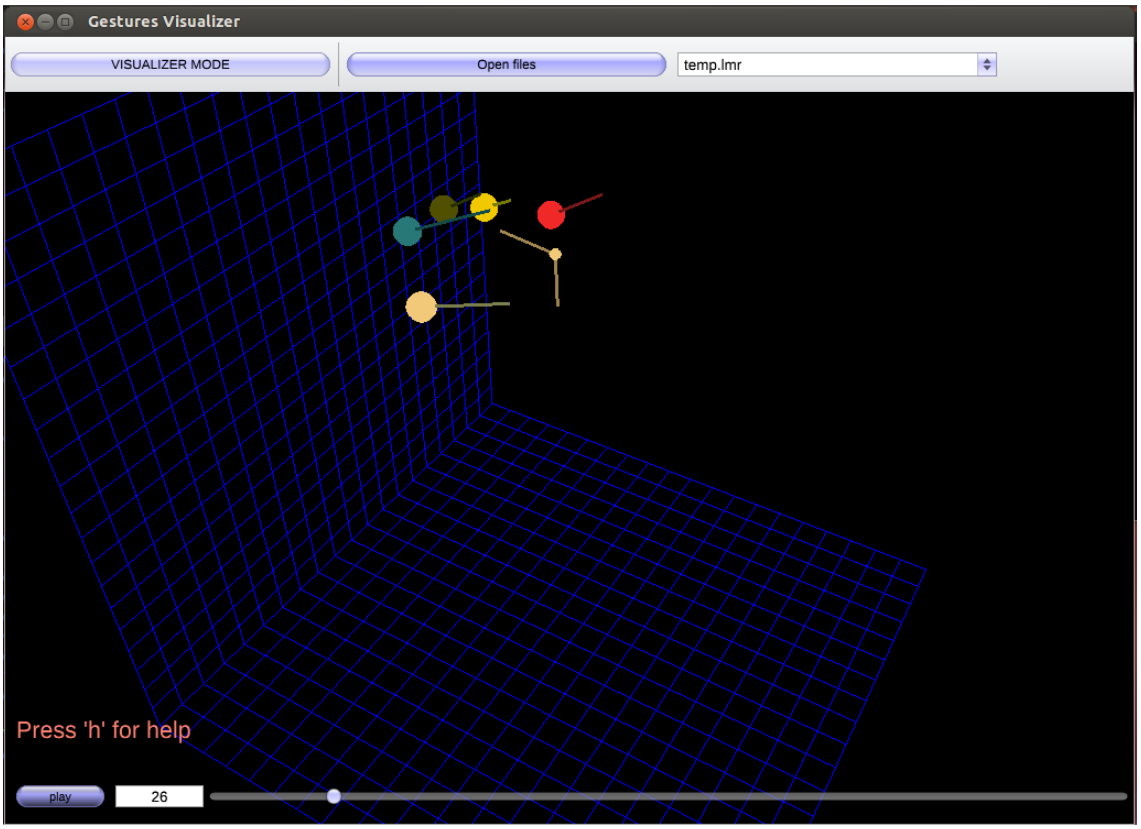

FIGURE 8.4: Screenshot of the visualizer

- The user has ability to rotate and zoom camera.
- Slider is available in order to move between frames of the recording.
- Visualizer has option to play recorded gesture, which the user can turn on and off by pressing the play/stop button.
- There is also a button to enter the recorder mode.

8.3.3 Recorder

Recorder is part of the described module, which is responsible for collecting information from Leap Motion Controller and saving it to LMR file.

Each frame read from the Leap Motion is captured, and then converted to the previously described model. Then the model is saved by the appropriate sub-module to LMR file. The conversion process contains also sorting hands and fingers. Hands are sorted by X coordinate of palmPosition. In the case of sorting fingers, the usual sort by the X coordinate is not enough. The order of the fingers must be independent of hand rotation, therefore a different method for fingers sorting had to be proposed. Fingers are sorted by distance between finger tip position and the plane perpendicular to the surface of the hand. The plane has to contain the direction vector of a hand. This plane can be determined using the palm position, the direction vector and a normalized normal vector of the hand. Below is the formula for the distance (d) between finger tip position and designated plane.

$$d = -(f - pp) \cdot (\hat{hd} \times \hat{hn})$$

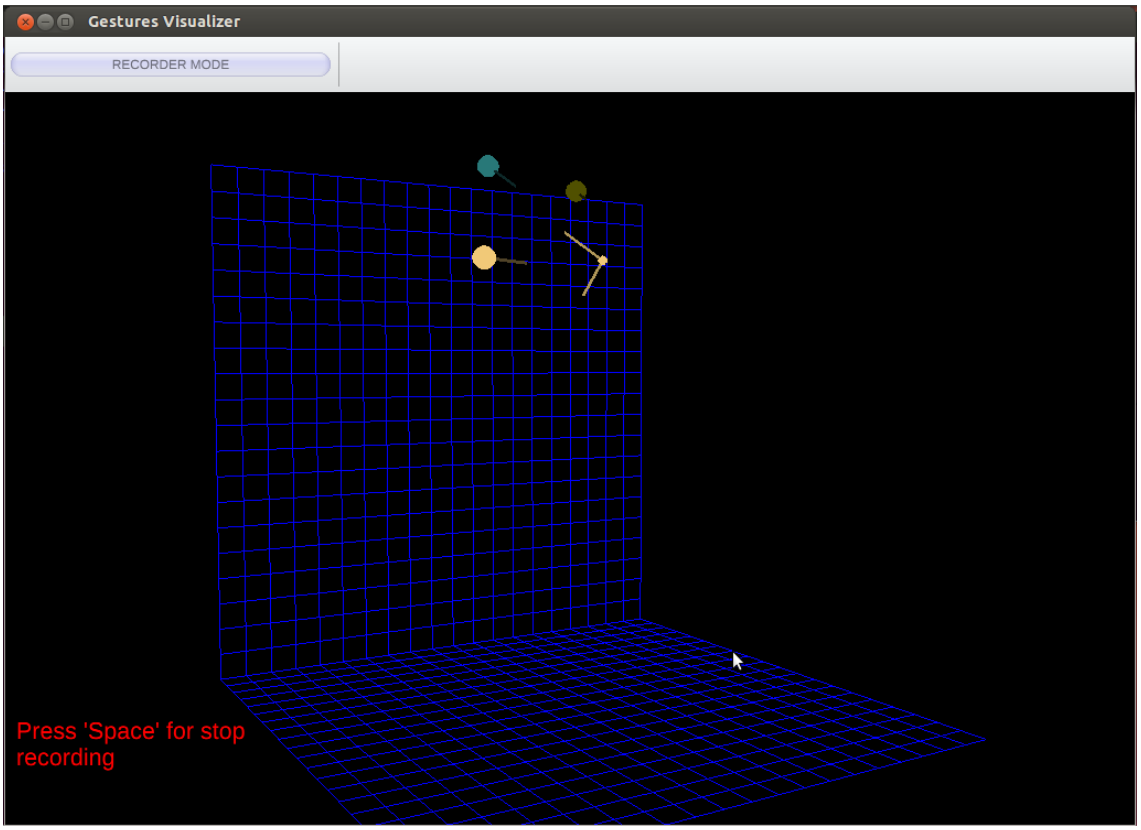

FIGURE 8.5: Screenshot of the recorder

Where:

f: is the finger tip position

pp: is the palm position

$\hat{hd}$: is a normalized hand direction vector

$\hat{hn}$: is a normalized hand normal vector

List of Recorder features:

- Turning on and off recording is done by using space key.
- After recording window automatically appears, in which can be choosen where to save the file.
- The recorder cooperates with the visualizer. During recording performed gesture is visualized.
- There is a button to enter the visualizer mode.

8.4 Used external libraries

- LeapSDK – allows to access Leap Motion device and its API
- pthread – library used to create lightweight processes (threads), to allow simulteanous (multithreaded) processing and provide task synchronization

- LIBSVM – "A Library for Support Vector Machines", provides SVM classifying abilities
- Dlib – provides linear algebra operations and machine learning ?
- HMMlib – a C++ library for general hidden Markov Models

8.5 Samples of code using the library dedicated to Leap Motion Controller

Here's an example of code required to use gesture recognition features.

```
1  // specifies method which will be executed after successful recognition
2  class MyGestures: public RecognizedGestureListener {
3
4    void onStaticRecognized() {
5      cout << "Recognized static gesture!" << endl;
6    }
7
8    void onDynamicRecognized() {
9      cout << "Recognized dynamic gesture!" << endl;
10   }
11
12 };
13
14 int main(int argc, char **argv) {
15
16   // create instance of recognized gesture actions
17   MyGestures *gst = new MyGestures();
18
19   // create new data processor
20   LeapProcess *process = new LeapProcess(gst, true, true);
21
22   // create a listener and attach it to a process
23   LeapListener listener;
24   listener.attachToProcess(process);
25
26   // create a Leap Motion controller object and direct its data output to a
         listener
27   Controller c(listener);
28
29   // start processing
30   process->start();
31
32   // as processing works in a separate thread, application can now perform other
         tasks
33   cin.get();
34
35   // stop processing
36   c.removeListener(listener);
37
38 }
```

Chapter 9

Conclusions

Bibliography

[1] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 12 1966.

[2] C. Biemann. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. *Proceedings of the HLT-NAACL-06 Workshop on Textgraphs-06*, 2006.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] Chin-Chen Chang, Jiann-Jone Chen, Wen-Kai Tai, and Chin-Chuan Han. New approach for static gesture recognition. *J. Inf. Sci. Eng.*, 22(5):1047–1057, 2006.

[6] Yen-Ting Chen and Kuo-Tsung Tseng. Developing a multiple-angle hand gesture recognition system for human machine interactions. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pages 489–492, 2007.

[7] Sven Teresniak Chris Biemann. *Disentangling from Babylonian Confusion – Unsupervised Language Identification*. Springer Berlin Heidelberg, 2005. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[8] Vladimir Vapnik Corinna Cortes. *Support-vector networks*.

[9] James Davis and Mubarak Shah. Visual gesture recognition, 1994.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[11] Fabio Dominio, Mauro Donadeo, Giulio Marin, Pietro Zanuttigh, and Guido Maria Cortelazzo. Hand gesture recognition with depth data. In *Proceedings of the 4th ACM/IEEE International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Stream*, ARTEMIS '13, pages 9–16, New York, NY, USA, 2013. ACM.

[12] Ahmed Elgammal, Vinay Shet, Yaser Yacoob, and Larry S. Davis. Learning dynamics for exemplar-based gesture recognition. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, pages 571–578, Washington, DC, USA, 2003. IEEE Computer Society.

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[14] Laurene Fausett, editor. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[15] Francisco Flórez, Juan Manuel García, and José García. Hand gesture recognition following the dynamics of a topology-preserving network. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR '02, pages 318–, Washington, DC, USA, 2002. IEEE Computer Society.

[16] Bartholomaus Rudak Frank Weichert, Daniel Bachmann and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors 2013, vol 5*, 2013.

[17] Nicholas Gillian and Joseph Paradiso. The gesture recognition toolkit. In *New England Machine Learning Day*, London, 2012.

[18] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[19] Haitham Hasan and S. Abdul-Kareem. Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, pages 1–35, 2012.

[20] Andreas Höfer, Aristotelis Hadjakos, and Max Mühlhäuser. Gyroscope-based conducting gesture recognition. In *NIME 2009 Proceedings: International Conference on New Interfaces for Musical Expression*, pages 175–176, 2009.

[21] Pengyu Hong, Matthew Turk, and Thomas S. Huang. Constructing finite state machines for fast gesture recognition. In *In Proc. 15th ICPR*, pages 691–694, 2000.

[22] Thomas S. Huang and Vladimir I. Pavlovic. Hand gesture modeling, analysis, and synthesis. In *In Proc. of IEEE International Workshop on Automatic Face and Gesture Recognition*, pages 73–79, 1995.

[23] M.B. Kaâniche. *Human Gesture Recognition*. 2009.

[24] Dietrich Kammer, Georg Freitag, Mandy Keck, and Markus Wacker. Taxonomy and overview of multi-touch frameworks: Architecture, scope and features. In *Workshop on Engineering Patterns for Multitouch Interfaces*, Berlin, 2010.

[25] Maria Karam and M. C. Schraefel. A taxonomy of gestures in human computer interactions. Technical report, 2005.

[26] N.Y.Y. Kevin, S. Ranganath, and D. Ghosh. Trajectory modeling in gesture recognition using cybergloves reg; and magnetic trackers. In *TENCON 2004. 2004 IEEE Region 10 Conference*, volume A, pages 571–574 Vol. 1, 2004.

[27] Jonghwa Kim, Stephan Mastnik, and Elisabeth André. Emg-based hand gesture recognition for realtime biosignal interfacing. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 30–39, New York, NY, USA, 2008. ACM.

[28] Davis E. King. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, 10:1755–1758, December 2009.

[29] Jiayang Liu, Zhen Wang, Lin Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–9, 2009.

[30] Yun Liu, Zhijie Gan, and Yu Sun. Static hand gesture recognition and its application based on support vector machines. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08. Ninth ACIS International Conference on*, pages 517–521, 2008.

[31] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[32] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:677–695, 1997.

[33] Matthew Peeples. R Script for K-Means Cluster Analysis. [on-line] `http://www.mattpeeples.net/kmeans.html`, 2011.

[34] Valery A Petrushin. Hidden markov models: Fundamentals and applications. In *Online Symposium for Electronics Engineer*, 2000.

[35] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, MobiCom '13, pages 27–38, New York, NY, USA, 2013. ACM.

[36] Francis Quek, David McNeill, Robert Bryll, Susan Duncan, Xin-Feng Ma, Cemil Kirbas, Karl E. McCullough, and Rashid Ansari. Multimodal human discourse: Gesture and speech. *ACM Trans. Comput.-Hum. Interact.*, 9(3):171–193, September 2002.

[37] L. Rabiner and B.-H. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[38] Md. Hafizur Rahman and Jinia Afrin. Hand gesture recognition using multiclass support vector machine. *International Journal of Computer Applications*, 74(1):39–43, July 2013. Published by Foundation of Computer Science, New York, USA.

[39] S. Rajko, Gang Qian, T. Ingalls, and J. James. Real-time gesture recognition with minimal training requirements and on-line learning. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[40] Yu Ren and Fengming Zhang. Hand gesture recognition based on meb-svm. In *Embedded Software and Systems, 2009. ICESS '09. International Conference on*, pages 344–349, 2009.

[41] Hrvoje Benko Michael Haller David Lindbauer Alexandra Ion Shengdong Zhao Roland Aigner, Daniel Wigdor and Jeffrey Tzu Kwan Valino Koh. Understanding mid-air hand gestures: A study of human preferences in usage of gesture types for hci. Technical report, November 2012.

[42] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(0):53 – 65, 1987.

[43] Alexandre Savaris and Aldo von Wangenheim. Comparative evaluation of static gesture recognition techniques based on nearest neighbor, neural networks and support vector machines. *J. Braz. Comp. Soc.*, 16(2):147–162, 2010.

[44] Xiaohui Shen, Gang Hua, Lance Williams, and Ying Wu. Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields. *Image Vision Comput.*, 30(3):227–235, March 2012.

[45] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 265–270, 1995.

[46] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804, 1957.

[47] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Eng. Appl. Artif. Intell.*, 22(8):1141–1158, December 2009.

[48] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Trans. Graph.*, 28(3):63:1–63:8, July 2009.

[49] Sy Bor Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1521–1527, 2006.

[50] Sabine Webel, Jens Keil, and Michael Zoellner. Multi-touch gestural interaction in x3d using hidden markov models. In Steven Feiner, Daniel Thalmann, Pascal Guitton, Bernd Fröhlich, Ernst Kruijff, and Martin Hachet, editors, *VRST*, pages 263–264. ACM, 2008.

[51] Ying Wu and Thomas S. Huang. Vision-based gesture recognition: A review. In *Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction*, GW '99, pages 103–115, London, UK, 1999. Springer-Verlag.

[52] Deyou Xu, Wuyun Yao, and Yongliang Zhang. Hand gesture interaction for virtual training of spg. In *ICAT Workshops*, pages 672–676. IEEE Computer Society, 2006.

[53] J. Yamato, Jun Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pages 379–385, 1992.

[54] Jie Yang and Yangsheng Xu. Hidden markov model for gesture recognition. Technical Report CMU-RI-TR-94-10, Robotics Institute, Pittsburgh, PA, May 1994.

[55] Jie Yang and Yangsheng Xu. Hidden markov model for gesture recognition. Technical report, DTIC Document, 1994.

[56] Ming-Hsuan Yang and N. Ahuja. Recognizing hand gesture using motion trajectories. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1, pages –472 Vol. 1, 1999.

[57] Pujan Ziaie, Thomas Müller, Mary Ellen Foster, and Alois Knoll. A naïve Bayes classifier with distance weighting for hand-gesture recognition. In *Advances in Computer Science and Engineering*, volume 6 of *Communications in Computer and Information Science*, pages 308–315. Springer Berlin Heidelberg, 2009.

© 2014 Michał Nowicki, Olgierd Pilarczyk, Jakub Wąsikowski, Katarzyna Zjawin

Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Typeset using LaTeX in Computer Modern.

BibTeX:

```
@mastersthesis{ key,
    author = "Michał Nowicki \and Olgierd Pilarczyk \and Jakub Wąsikowski \and Katarzyna Zjawin",
    title = "{Gesture recognition library for Leap Motion Controller}",
    school = "Poznan University of Technology",
    address = "Pozna{\'n}, Poland",
    year = "2014",
}
```