Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Bachelor's thesis

BIBLIOTEKA DO ROZPOZNAWANIA GESTÓW DLA KONTROLERA LEAP MOTION

Katarzyna Zjawin, xxx
Michał Nowicki, 95883
Olgierd Pilarczyk, xxx
Jakub Wąsikowski, xxx

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

Static gestures recognition

As was already mentioned, the detected gestures can be divided into two groups: static gestures and dynamic gestures. The static gestures can be understood as a chosen position and orientation of the fingers and hand in a single moment, while dynamic gestures are defined as a movement of the hand and fingers in time. The problem of recognition of those gestures is a subject of following chapter. Firstly, the proposed approach is presented, followed by the introduction to the evaluation scheme. In last section, the performed experiments are descriebed, which were used to examine the effectiveness of proposed static gesture recognition approach.

1.1 Proposed methods

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

While for most 2D gesture recognition problems simple classification algorithms seems to work well enough, the 3D data is more complication to model by the set of features and finally successfully label. While dealing with 3D data, the position and orientation of hand can be easily affected by the height of the hand above the sensor or small change in the orientation of the hand with respect to the sensor's coordinate system. It is intuitively understood, that the system should recognize those gestures as the one as they are similar. To meet those requirement, one need to define what is meant by the „small" change in orientation resulting in treating the static gestures as the same.

To meet those requirements, the Support Vector Machines [CC] were used as a classification algorithm. The SVMs were chosen as there exist a solid mathematical background supporting the simple idea of maximalizing the margin between classes. Moreover, the SVMs were chosen also because of the popularity due to the open-source library libSVM [CL11], which contains the multiple platform SVM implementation. It is worth noticing that in original work, SVMs were used only to classify between two classes, but the idea was expanded to utilize the one-vs-all scheme allowing to classify multiple class sets. The efficiency of SVMs depends on correctly choosing the kernel function used to map the separation problem into higher-dimension with expectation to achieve problem easier to solve. The typical kernel functions:

- linear: $K(x_i, x_j) = x_i^T x_j$.
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.

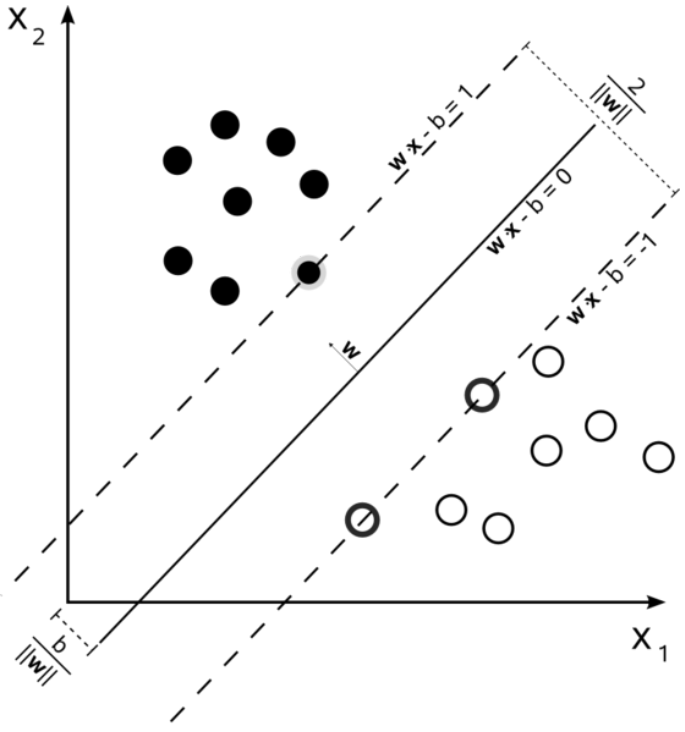

FIGURE 1.1: SVM is a technique searching for the hiperplane that maximizes the margin between classes[1]

- 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, linear kernels should be used for linearly separable problems, while RBF kernel is the most versatile one.

The problem of classification assumes that each sample consists of set of features, which describe this sample and can be used to distinguish it from the other samples. Additionally, each sample has a known or unknown label, which defined the membership of sample to the class. The samples with the known labels can be used to train the classification system to compute the membership to the classes for the samples. The computation is performed on previously mentioned sets of features.

In application of gesture recognition the classification be divided into two flows: the training part and the recognition part. In training part, the library will be provided with the samples of static gestures with known correspondences to the static gesture classes. 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 with samples static gestures without labels. For each sample the sets of features are computed and then given as input to the trained classifier. The classifier returns the information of the gesture's class membership (label) of each sample.

In case of library, it is assumed that the learning process can be done offline, while strict online requirements has to be met in recognition part. To meet those requirements the Support Vector Machine is introduced[CC]. The SVM classification is commonly used technique in multiple areas of research as biology, robotics or IT for solving data classification problems. Additional advantage

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

of the SVM is possibility to use C++ library libSVM [CL11], which provides an easy interface to utilize this classification methods in different problems.

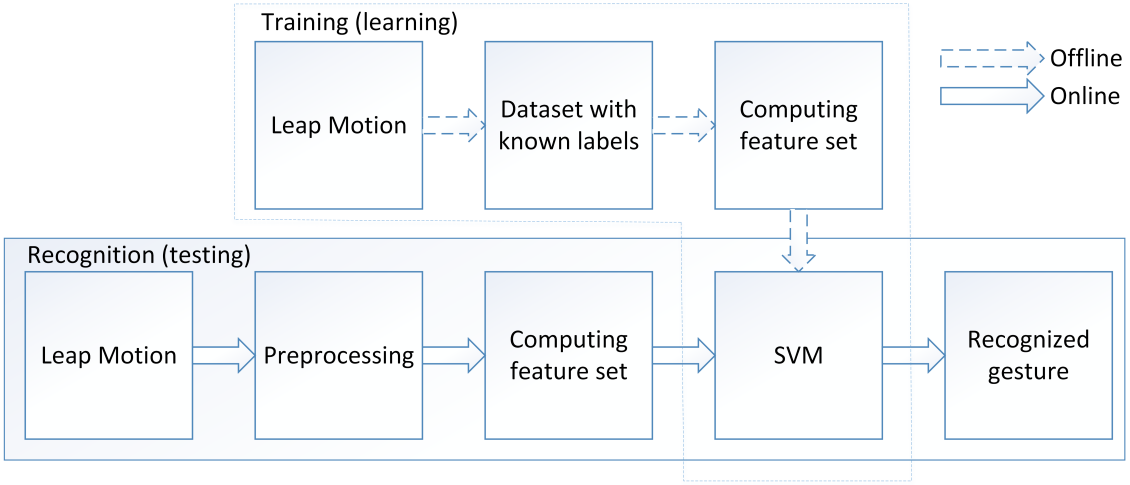

FIGURE 1.2: Proposed solution blocks for learning and recognition parts of static gestures recognition problem

While 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 proposed features were dependent on the position, orientation and scale of hand. Even small movement in any direction meant problems with stable recognition. The theoretical literature suggests to compute a set of features invariant to wanted transformations, which can allow to fully distinguish between different classes. Unfortunately, there are not available propositions to feature sets when it comes to the gesture recognition using the data even similar to the data provided by the Leap Motion sensor. Seeking right feature is a task undertaken in experimental section 1.3.

To sum up, the static gesture processing flow is presented at fig. 1.2. In training part, the data from Leap Motion is preprocessed, the feature sets are computed and the data is used to train SVM classifier. In recognition part, the data is also preprocessed and described by the feature sets, but the knowledge of the label comes from the already trained SVM classifier.

1.2 Evaluation methodology

1.2.1 Assumptions

To provide user with library working in different conditions, it was assumed that the 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 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 feasibility of the library. It is worth mentioning that, it does not reduce the possible applications of the library, as an assignment of static gesture to already defined class allows to find the transformation between the model of the class and observed gesture.

1.2.2 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. American Sign Language: "I love you" sign,
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.

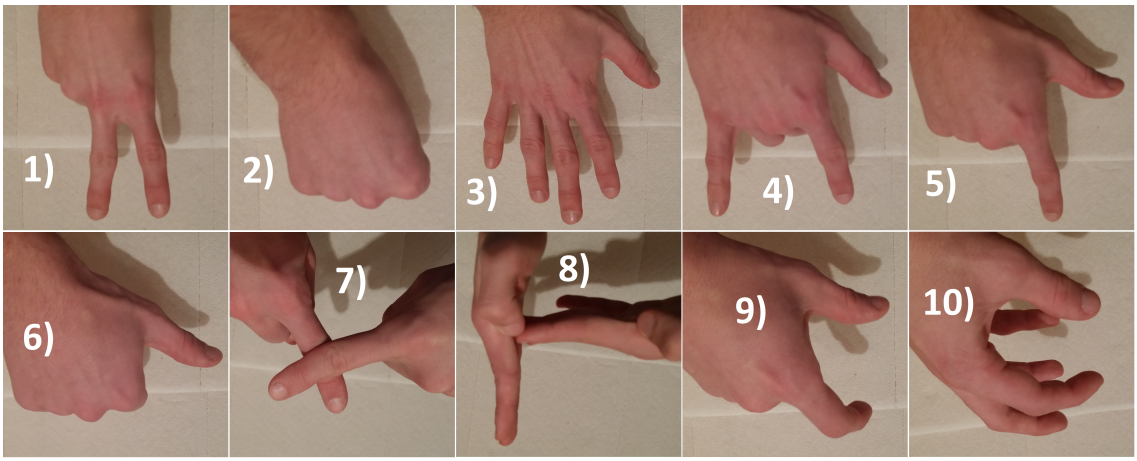

Figure 1.3: Recorded static gestures used for evaulation

The gestures are also presented at fig. 1.3.

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. For each of the proposed gestures, each author 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 trying to prevent the algorithm from over-fitting the training data. This method is used to find the optimal parameters of the classification system, while estimating the performance on the data not used in the training part. In standard version of the method, the gathered data is divided into two sets: one containing k-1 parts of the data, 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 the performance. The performance is estimated by calculating the number of cases when the classification system returned a label which matched already known

label. The percent of correctly recognized labels to the total size of the testing set is known as recognition rate.

1.3 Experiments

Firstly, the experiments to find the proper set of features were conducted. The first proposed vector of features consisted of:

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

While this feature set did not take into account the relative position of fingers to the hand, the second and third feature set were introduced. The second feature set is a 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 second feature set extended by the five angles between fingers and normal of hand's palm. Those, 3 proposed feature sets were firstly tested on all 10 recorded static gestures. 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 how the Leap Motion works. For gestures like fist or 'X' 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 the table 1.3.

Table 1.1: Results obtained by experimental feature sets by the libSVM library

	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
feature set 1	87.072%	87.943%	69.987%	68.422%
feature set 2	87.072%	87.943%	69.987%	68.422%
feature set 3	87.072%	87.943%	69.987%	68.422%
feature set 4	81.150%	80.998%	68.438%	68.630%
feature set 5	86.544%	85.075%	77.101%	77.518%
feature set 6	92.762%	93.096%	80.543%	81.235%

For problem of recognition of 5 gestures, 3 first feature sets resulted in over 87% recognition rate on testing sets, which was believed to be a good result. The same tests for 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 and revealed that the fingers are numbered accordingly to the position in Z axis of the tip of the finger. This means that when fingers' 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 yet again

presented in table 1.3. 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 gesture problem in case of feature set 6.

TABLE 1.2: Results obtained by experimental feature sets by the libLinear library

	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
feature set 1	78.282%	78.283%	50.596%	50.593%
feature set 2	78.282%	78.283%	50.596%	50.593%
feature set 3	78.282%	78.283%	50.596%	50.593%
feature set 4	78.205%	78.242%	50.658%	50.575%
feature set 5	79.527%	79.502%	55.284%	55.263%
feature set 6	88.0756%	88.138%	64.747%	64.830%

While using more data and longer feature sets usually allows to achieve better results, it is worth to notice the growth of training time of classification technique. In case of 5000 training samples the typical training process of radial SVM took approximately 12 hours on standard desktop PC. This computing time can be unacceptable by the users of the library, so the test with another SVM library libLinear [FCH$^+$08] was performed. The libLinear's implementation of SVM utilizes the linear kernels, which are useful 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 tab. 1.3. For the 5 gesture case, the libLinear achieved 88.138% on testing set while using feature set 6 compared to the 93.096% achieved by libSVM in the same condition. In this case, the libLinear might be good choice as the recognition rate different is about 5%. For 10 gesture case, libLinear achieved 64.830% compared to 81.235% by LibSVM, which is a significantly lower. It's up to user to decide, which library to use, but in most recognition taks applications the library can be learnt offline and only used online. That is why, the SVM with RBF kernels is a recommended choice in 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.

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 rate. The preprocessing operates in the window of hands poses recorded over time, which size can be modified. The typical library usage allowed to gather data with $60Hz$, while it is assumed that the recognition can be performed with lower framerate. The difference can be efficiently used by defining the appropriate preprocessing window. For these 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 tab. 1.3. 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 93.096% to over 99% for window sizes equal or wider than 15. In task of correctly detecting 10 gestures, the preprocessing allowed to increase recognition rate from 81.235% 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 further experiments.

TABLE 1.3: The recognition rate achieved with feature set 6 and different parameters of preprocessing

preprocessing	5 gestures, CV	5 gestures, test set	10 gestures, CV	10 gestures, test set
off	92.762%	93,096%	80.543%	81.235%
width = 2	95.641%	96,527%	82.736%	83.172%
width = 5	95.861%	96,650%	83.078%	83.565%
width = 10	98.796%	98,965%	83.602%	84.071%
width = 15	98.981%	99,150%	84.112%	84.494%
width = 20	99.104%	99,242%	84.329%	84.834%
width = 30	99.232%	99,385%	84.848%	85.320%

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 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. Firstly, it is important to use not only the class labels for 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 rate for each class in a window of set 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:

$$l(i) = [l_{i1}, l_{i2}, ..., l_{ik}] \tag{1.1}$$

where l_{ik} is the likelihood of belonging of i-th data to the k-th class. Then combining in a window of width w can be written as:

$$r(i, w) = \sum_{j=i-w}^{i} l(j) \tag{1.2}$$

where the sum of vectors l can be defined freely. The recognized label is the number of the vector component with the highest value:

$$\text{label} = \arg\max_{k} \{r(i, w)_k\} \tag{1.3}$$

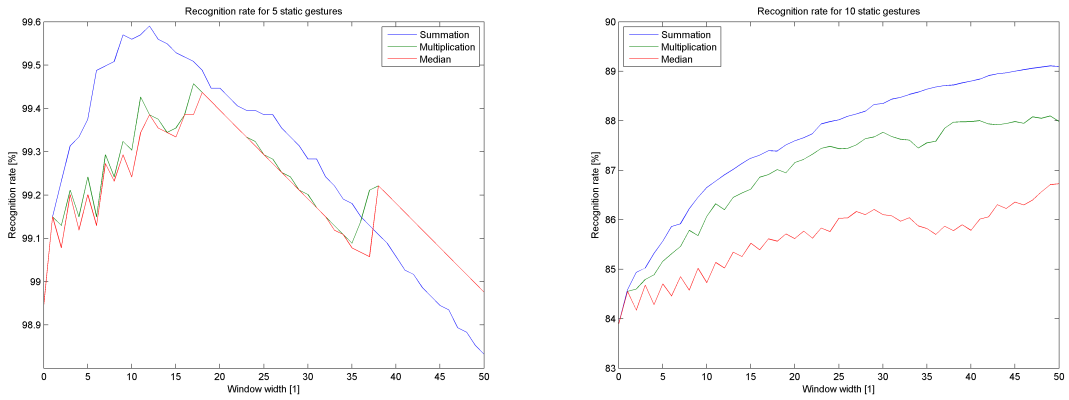

FIGURE 1.4: Evaluation of different operators for result combining in case of different window sizes

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 for 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 at fig. 1.4. For almost all possible widths, the summation operator demonstrated the best recognition rate. For 5 static gesture, 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 best achieved with window size equal to 10. For 10 static gesture recognition problem, using window of size equal to 50 allowed to increase 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 will result in delayed recognition rate of shown gesture. Also, too wide window may result in worse results. Therefore, the postprocessing window size of 10-15 is recommended, but it also depends on the application of the library.

With simple summation window, the currently achieved result and the results from the past are equally affecting the recognition result. 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 weight distribution should have the highest weight for the current measurement and smaller values for results that were achieved earlier in time.

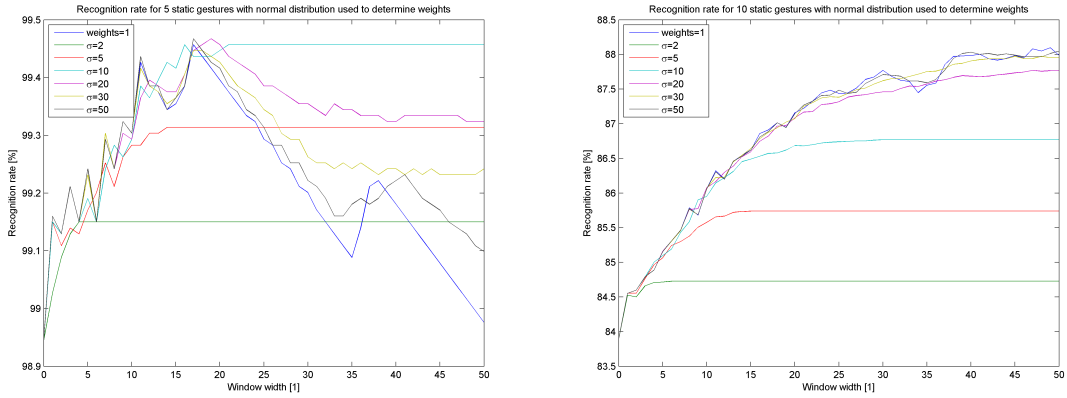

FIGURE 1.5: Evaluation of weights for postprocessing distributed accordingly to normal distribution

For this task, the half of the Gaussian distribution of weights can be used with maximal peak reached for the currently achieved result with weights slowly decreasing for measurements further away in time. For Gaussian, the mean was assumed to be equal to 0. The standard deviation σ was the parameter, which different values were tested. The results were also performed for problems of recognition of 5 and 10 gestures. The achieved results are presented in fig. 1.5. For 5 gestures, too small σ prevented the increase of recognition rate due to the wider window size as the weights were equal to 0. For greater sigmas, the achieved recognition rate is similar. For this problem, the σ equal to 10 resulted in best recognition rate while the window was getting wider. For problem of 10 static gesture recognition problem, greater sigma values achieved results comparable with the the results achieved with weights equal to 1. As the usage of different weights also comes with greater computing cost and still did not result in better results, this part of processing can omitted without effecting the recognition.

From the presented results, the simple summation with uniform weights provides the greatest gain when it comes to the recognition rate and is the method recommended by the authors.

Chapter 2

Detection of dynamic gestures

2.1 Proposed methods

The dynamic gesture recognition problem is a problem, where the input data consist of several consecutive positions and orientations of hand and fingers. Moreover, the important factor for recognition is the time dependencies between data frames. The slower and faster gestures 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 same features as in the static recognition part. The set of features for each frame is then processed by the Hidden Markov Model scheme.

2.1.1 Hidden Markov Model

Hidden Markov Model is model of a system with Markov property. The first introduction of HMM comes from the L. E. Baum et. al.[BP66], who proposed the mathematical background for HMMs. A HMM can be considered a finite, N-element set of states, which are associated with the probability distribution. 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. To fully define the model, the definition of N-element vector of initial state probabilities Π. Each HMM can be fully defined by the (T, E, Pi).

The example HMM can be seen at fig. **??**. The HMM at 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. The edges between states contain and are an equivalent to the transition matrix. There are no edges between vertices representing observations. The edges between states and observations are equal to the observation matrix.

xx There are three main algorithms used with the HMMs:

- Forward-Backward algorithm,
- Viterbi algorithm,

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

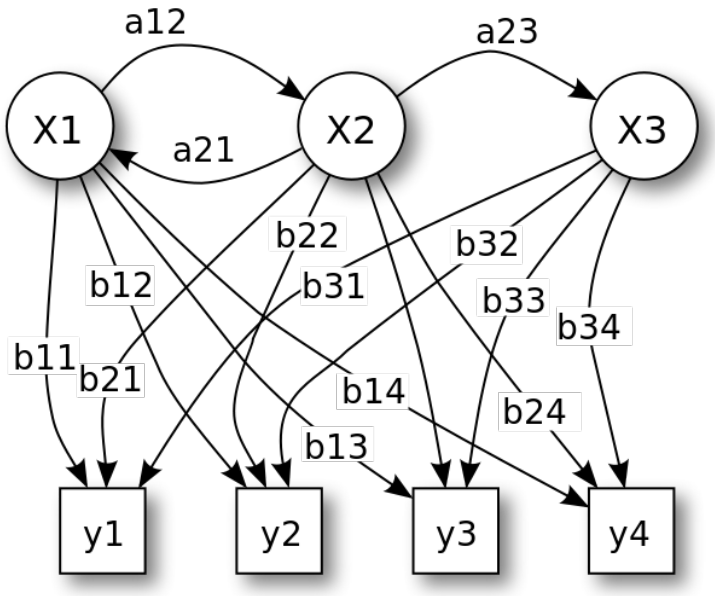

FIGURE 2.1: Solution blocks of learning and testing parts in task of dynamic gesture recognition[1]

- Baum-Welch algorithm.

The Forward-Backward algorithm is used to find the posterior probability of given states given the set of observations. For the whole set of state variables $X = X_{i\,i=1}^{N}$ and a set of observations $o_{1:t} = o_1, o_2, ..., o_t$, the algorithm computes the $P(X_i|o_{1:t})$. The algorithm utilizes a dynamic programming approach, by perfoming 3 steps in a loop:

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

Firstly, the forward pass phase for $i = 1, .., N$ computes the $P(X_i, o_{1:k})$, 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 the $(P(o_{k+1:t})|X_i)$, which are the probabilities of observing the rest of the observations in state X_i. The smooting part, uses Bayes rule to compute the probability of state X_i given the whole observation sequence:

$$P(X_k|o_{1:t}) = P(X_k|o_{1:k}, o_{k+1:t}) \propto P(o_{k+1:t}|X_k)P(X_k|o_{1:k}) \tag{2.1}$$

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.

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 usually called the Viterbi path. Introduing the sequence:

$$V_{1,k} = P(o_1|k) * \Pi_k) \tag{2.2}$$

$$V_{t,k} = P(o_t|k) * argmax_{x \in X}(a_{x,k} * V_{t-1,x}) \tag{2.3}$$

Finally, the problem can be understood as finding the $argmax_{x \in X}(V_{T,x})$. The algorithm time complexity is the same as forward-backward $O(N^2T)$.

The Baum-Welch algorithm is the algorithm used to find the unknown parameters of the HMM. For each training example, the algorithms updates the state transition probabilities, observation probabilities in each state and initial probabilities to maximize the likelihood of observation. Having the set of sets of observations, the algorithm can be used to train HMM to detect the sequences similar to the ones used in learning process.

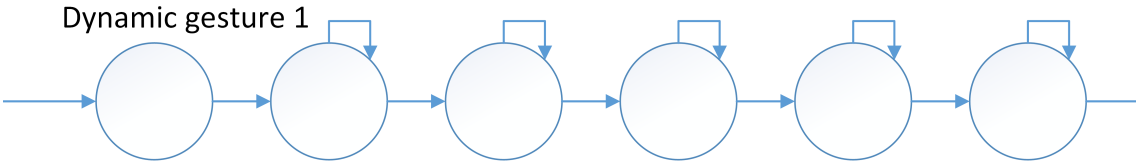

FIGURE 2.2: Non-zero state transitions and states of the HMM used to detect single dynamic gesture

In the dynamic gesture recognition task, we adopted a structure of HMM with state having non-zero transition probabilities to self and to the next state in the sequence. The proposed structure is presented at fig. 2.2 and was firstly proposed in paper [YX94]. The states can be understood as the phases of hand movement and position that happen when the wanted gesture is performed. The self-transitions are used to model the different speeds of the gestures. This structure after training process can be used to measure the probability of that the dynamic gesture occured given the set of observations.

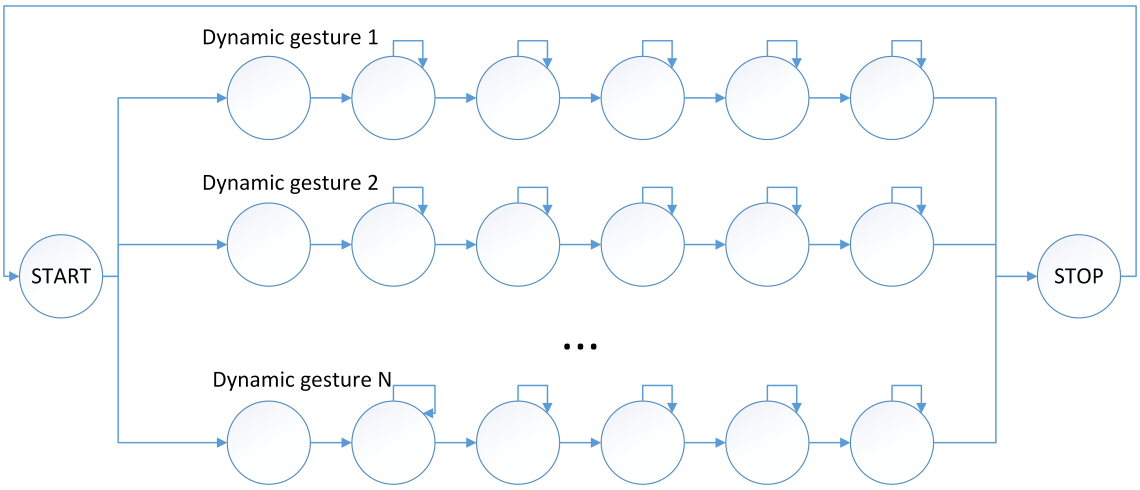

FIGURE 2.3: Non-zero state transitions and states for the HMM enabling simultaneous detection of m dynamic gestures

Having single dynamic gesture recognition problem modelled as the sequence of n states in which $kths$ state is connected by the edges to the k and $(k+1)$ state and to the all observations. Having the problem of distinguishing m gestures translates to the m sequential graphs. The problem of finding if and what dyanmic gesture occured is the problem of finding the probabilities from each m HMM and checking them against preset threshold. 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 at fig. 2.3. Then the recognitnion process is a process of finding, which of the parallel paths is the most probable. The only remaining problem is the problem of preparing the set of ccured, 1-dimensional observations from the Leap Motion data.

2.1.2 HMM observation from Leap Motion data

Most of the proposed solutions in the literature assume that the observations are 1-dimensional. There were attempts to expand the HMM to the 2-dimensional observations [], but the raw data from Leap Motion is defined in higher-dimensional that any found solution. That is the reason, why the need 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 [Bie06, CB05],
- Girvan-Newman clustering [GN02],
- k-means with additional methods to determine the number of clusters. [Ste57, Mac67].

Chinese whispers algorithm is an efficient randomized graph-clustering algorithm, which is time-linear in the number of edges. The algorithm was proposed in [Bie06] 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[Kin09], which provides multiple 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 the error condition is met. The solution iteratively calculates the eigenvalues of a created matrix using the power iteration method. In case of a complete graph, this algorithm is relatively slow.

The another proposed approach is a k-means clustering algorithm. The idea for the algorithm comes from polish mathematician Hugo Steinhaus [Ste57], but the first practical application has been presented in [Mac67]. The algorithms initially randomly selects k starting points and performs iteratively 2 steps utilizing the idea of expected-maximization [DLR77]:

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 maximum number of iterations or when the change between consecutive iterations is smaller then defined epsilon. Unfortunately, the algorithm requires the 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 [sil]. 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:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.4}$$

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.

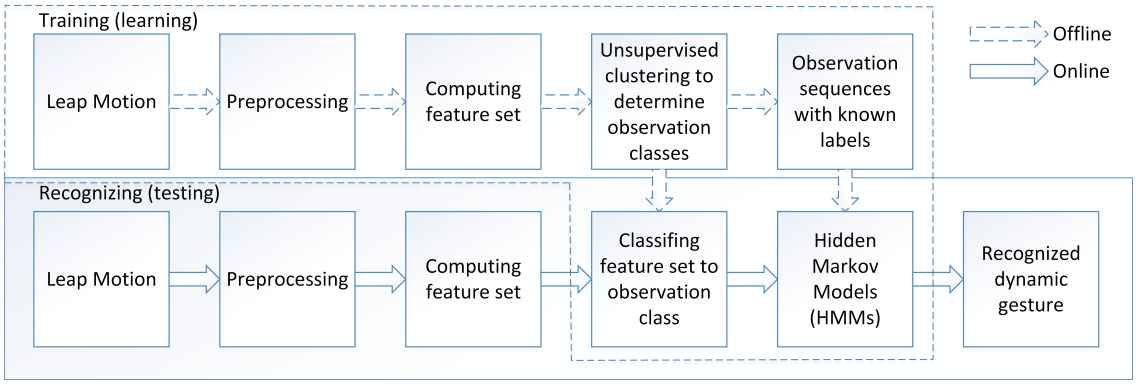

FIGURE 2.4: Solution blocks of learning and testing parts in task of dynamic gesture recognition

Finally, the whole processing flow has been designed and is presented at fig. 2.4. The whole solution consists of two parts: offline learning and online 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. To complete, the training process, the HMMs are combined into one HMM.

In the online 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 preset threshold, the gesture is assumed to be correctly recognized.

2.2 Evaluation methodology

To evaluate the quality of recognition achieved by proposed processing pipeline, 8 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 in different positions, with different speed to measure the wanted invariance. The chosen dynamic gestures are:

1. „number 8” – it's a number 8 made by one finger in the air,
2. „letter Z” – it's a letter Z made by one finger in the air,
3. open hand with 5 fingers going down in the direction of the sensor,
4. „clearing” – open hand moving left and right and simulating the movement of clearing the screen,

5. circle made by one finger in the air
6. „letter A" – a letter A made by one finger in the air,
7. „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.
8. „swipe from right" – moving the hand from right to left, used e.g. in smartphones to move something out of the current view.

To test the proposed approach, similarly to the static gesture recognition problem, the $120 \times 8 = 960$ samples were divided into training and testing group. 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. To prevent overfitting of the data and achieve the measure when the training process should be stopped, the k-fold cross validation was utilized.

2.3 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:

$$d(x, y) = \sqrt{\sum_{i=1}^{16} (x_i - y_i)^2} \tag{2.5}$$

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

$$s(x, y) = \frac{1.0}{\max\{d(x, y), eps\}} \tag{2.6}$$

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 set to $1e-4$. For the SSE analysis for different cluster number, the publicly available script was used[Pee11]. 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 toll provides several plots, but the most interesting one is presented at fig. 2.5. 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 proposed values of clusters for different methods are presented in the table 2.1.

********8 *********88 *DYNAMICZNE TESTY ********88 ********

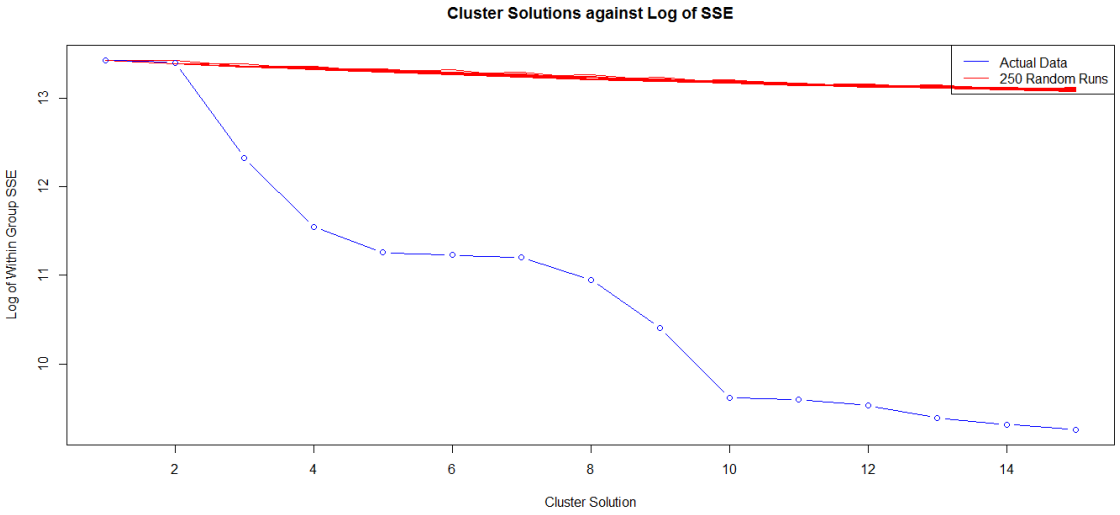

FIGURE 2.5: Different representations of SSE as a function of number of clusters for the k-means algorithm

TABLE 2.1: 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	x	xx	5 or 10	xx

Bibliography

[Bie06] 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.

[BP66] 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.

[CB05] 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`.

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

[CL11] 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`.

[DLR77] 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.

[FCH+08] 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.

[GN02] 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.

[Kin09] Davis E. King. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, 10:1755–1758, December 2009.

[Mac67] 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.

[Pee11] Matthew Peeples. R Script for K-Means Cluster Analysis. [on-line] `http://www.mattpeeples.net/kmeans.html`, 2011.

[sil]

[Ste57] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804, 1957.

[YX94] Jie Yang and Yangsheng Xu. Hidden markov model for gesture recognition. Technical report, DTIC Document, 1994.

Poznan University of Technology
Faculty of Computing
Institute of Computing Science

Typeset using LaTeX in Computer Modern.

BibTeX:

```
@mastersthesis{ key,
    author = "Katarzyna Zjawin \and Michał Nowicki \and Olgierd Pilarczyk \and Jakub Wąsikowski",
    title = "{Biblioteka do rozpoznawania gestów dla kontrolera Leap Motion}",
    school = "Poznan University of Technology",
    address = "Pozna{\'n}, Poland",
    year = "2014",
}
```