Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

One-Shot-Learning Gesture Recognition using HOG-HOF Features

Bachelor Thesis

2013

Jakub Konečný

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

One-Shot-Learning Gesture Recognition using HOG-HOF Features

Bachelor Thesis

Study programme: Mathematics of Economy and Finance
Study field: 9.1.9. Applied Mathematics (1114)
Department: Department of Applied Mathematics and Statistics
Supervisor: doc. Mgr. Radoslav Harman, PhD.

Bratislava, 2013
Jakub Konečný

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jakub Konečný
Študijný program: ekonomická a finančná matematika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.1.9. aplikovaná matematika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický

Názov: One-Shot-Learning Gesture Recognition using HOG-HOF Features

Cieľ: Create a learning system capable of learning from a single training example a gesture classification problem using data obtained from Microsoft Kinect. Practice with development data (a large database of 50,000 labeled gestures is available).

Vedúci: doc. Mgr. Radoslav Harman, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, CSc.

Dátum zadania: 10.10.2012

Dátum schválenia: 03.11.2012 doc. RNDr. Margaréta Halická, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgement

I would like to thank my supervisor doc. Mgr. Radoslav Harman, PhD. for his help.

Abstract

The purpose of this thesis is to describe one-shot-learning gesture recognition systems developed on the *ChaLearn Gesture Dataset* [3]. We use RGB and depth images and combine appearance (Histograms of Oriented Gradients) and motion descriptors (Histogram of Optical Flow) for parallel temporal segmentation and recognition. The Quadratic-Chi distance family is used to measure differences between histograms to capture cross-bin relationships. We also propose a new algorithm for trimming videos — to remove all the unimportant frames from videos. Our two methods both outperform other published methods and help narrow down the gap between human performance and algorithms on this task. The code has been made publicly available in the MLOSS repository.

Keywords: ChaLearn, Histogram of Oriented Gradients, Histogram of Optical Flow, Dynamic Time Warping

Abstrakt

Cieľ tejto práce je popísať systémy na rozpoznávanie gest na základe jedného trénovacieho príkladu, vytvorené pre *ChaLearn Gesture Dataset* [3]. Používame farebný aj hĺbkový obraz a kombinujeme statické (Histogram of Oriented Gradients) a dynamické (Histogram of Optical Flow) deskriptory na paralelné rozpoznávanie gest a segmentáciu v čase. Na meranie vzdialeností medzi histogramami sme použili metriku Quadratic-Chi aby sme zachytili vzťahy medzi jednotlivými bunkami. Taktiež predstavujeme nový algoritmus na orezávanie videí, aby sme odstránili všetky nepodstatné snímky z videa. Naše obe metódy prekonávajú iné publikované metódy a pomohli zmenšiť rozdiel medzi výkonom ľudí a algoritmov v tejto úlohe. Zdrojový kód sme zverejnili v databáze MLOSS.

Kľúčové slová: ChaLearn, Histogram of Oriented Gradients, Histogram of Optical Flow, Dynamic Time Warping

Contents

Introduction

Gesture recognition can be seen as a way for computers to understand human body language. Improving state-of-the-art algorithms for gesture recognition thus facilitates the human-computer communication beyond primitive text user interfaces or GUIs (graphical user interfaces). With rapidly improving comprehension of human gestures we can start building NUIs (natural user interfaces) for controlling computers or robots. With the availability of such technologies, conventional input devices like keyboard or mouse could be replaced in situations in which they are inconvenient in future. Other applications of gesture recognition include sign language recognition, socially assistive robotics and astonishing game technology.

In this thesis, we focus on the *ChaLearn Gesture Dataset* [3]. The dataset was released jointly with a competition, which purpose was to develop a system capable of learning to recognize new categories of gestures from a single training example of each gesture. The large dataset of hand and arm gestures was pre-recorded using an infrared sensor, Kinect^{TM}, providing both RGB and depth images [8, 7].

The purpose of this work is to describe methods developed during the *ChaLearn Gesture Challenge* by team Turtle Tamers (author of this thesis and Michal Hagara). The thesis itself is based on our paper submitted to the Journal of Machine Learning Research. We finished in the 2^{nd} place in round 2 and were invited to present our solution at the International Conference on Pattern Recognition 2012, Tsukuba, Japan. The code has been made publicly available in the MLOSS repository.[1]

The rest of this work is organised as follows: Related work is summarized in Chapter 1. In Chapter 2 we describe the dataset and the problem thoroughly. In Chapter 3 we focus on the preprocessing needed to deal with some of the problems in the dataset. Chapter 4 covers feature representation, using Histogram of Oriented Gradients and Histogram of Optical Flow, as well as a method used to compare similarities between these representations. In

[1] https://mloss.org/software/view/448/

Chapter 5 we describe the actual algorithms, and in Chapter 6 we briefly describe algorithms of other participants and compare their results with ours, as well as with other published works. We finish by concluding in Chapter 7.

Chapter 1

Related Work

In this section we highlight selected works in the area of action recognition and motivate our choices of models.

One possible approach to this problem consists in analysing motion descriptors obtained from video. For example [10] use output of Human Motion Capture systems with combinations of Hidden Markov Models. Authors of [23] use Extended Motion History Image as a motion descriptor and applies the method on the *ChaLearn Gesture Dataset*. They fuse duo modalities inherent in the Kinect sensor using Multiview Spectral Embedding [24] in a physically meaningful manner.

An evolution of Bag-of-Words [15], a method used in document analysis, where each document is represented using the apparition frequency of each word in a dictionary, is one of the most popular in Computer Vision. In the image domain, these words become visual elements of a certain visual vocabulary. First, each image is decomposed into large set of patches, obtaining a numeric descriptor. This can be done using for example SIFT [16], or SURF [1]. A set of N representative visual words are selected by means of a clustering process over the descriptors in all images. Once the visual vocabulary is defined, each image can be represented by a global histogram containing the frequencies of visual words. Finally, this histogram can be used as input for any classification technique. Extensions to image sequences have been proposed, the most popular being Space-Time Interest Points [13]. An evaluation of a number of feature descriptors and bag-of-feature models for action recognition is presented in [22]. This study concluded that different sampling strategies and feature descriptors were needed to achieve the best results on alternative action data sets. Recently an extension of these models to the RGB-D images, with a new depth descriptor was introduced by [9].

These above outlined methods usually ignore particular spatial position of a descriptor. We wanted to exploit the specifics of the dataset, particularly the fact that the user position does not change within the same batch, thus also the important parts of the same gestures will occur roughly at the same place. We use a combination of appearance descriptor, Histogram of Oriented Gradients [4] and local motion direction descriptor, Histogram of Optical Flow [11]. We adopted Quadratic-Chi distance [20] to measure differences between these histograms. This approach works well only at high resolutions of descriptors. An alternative to this could be to use a non-linear support vector machine with a χ^2 kernel [14]. Another possible feature descriptor that includes spatio-temporal position of features could be HOG3D [12], which was applied to this specific dataset by [6].

Chapter 2

Data and Problem Setting

In this Section, we will discuss the easy and difficult aspects of the dataset and the goal of the competition.

As mentioned earlier, the purpose of the *ChaLearn Gesture Challenge*[1] was to develop a system capable of learning to recognize new categories of gestures from a single training example of each gesture. A large dataset of gestures was collected before the competition, which includes more than $50,000$ gestures recorded with the Kinect^{TM} sensor, providing both RGB and depth videos. Resolution of these videos is 240×320 pixels, at 10 frames per second. The gestures are grouped in more than 500 batches of 100 gestures, each batch including 47 sequences of 1 to 5 gestures drawn from small gesture vocabularies from 8 to 14 gestures. The gestures come from over 30 different gesture vocabularies, and were performed by 20 different users.

During the challenge, development batches devel01-480 were available, with truth labels of gestures provided. Batches valid01-20 and final21-40 were provided with labels for only one example of each gesture class in each batch (training set). These batches were used for evaluation purposes. The goal is to automatically predict the gesture labels for the unlabelled gesture sequences (test set). The gesture vocabularies were selected from nine categories corresponding to various settings or applications:

1. body language gestures (for example scratching your head, crossing your arms)
2. gesticulations performed to accompany speech

[1]The challenge was organized by ChaLearn and sponsored in part by Microsoft (Kinect for Xbox 360). The submission website was hosted by Kaggle.com. Other sponsors include Texas Instrument. This effort was initiated by the DARPA Deep Learning program and was supported by the US National Science Foundation (NSF) under grants ECCS 1128436 and ECCS 1128296 , the EU Pascal2 network of excellence and the Challenges in Machine Learning foundation. Website: http://gesture.chalearn.org/.

3. illustrators (like Italian gestures)
4. emblems (like Indian Mudras)
5. signs (from sign languages for the deaf)
6. signals (like referee signals, diving signals, or marshalling signals to guide machinery or vehicle)
7. actions (like drinking or writing)
8. pantomimes (gestures made to mimic actions)
9. dance postures

Easy aspects of the dataset include fixed camera, availability of the depth data. Within a batch, there is always a single user, homogeneous recording conditions and a small vocabulary. In every sequence, different gestures are separated by the user returning to a resting position. Gestures are usually performed by hands and arms. The challenging aspects of the data are that within a single batch there is only one labelled example of each gesture. Between different batches there are variations in recording conditions, clothing, skin color and lightning. Some users are less skilled than others, thus there are some errors or omissions in performing the gestures. And in some batches, parts of the body may be occluded.

For the evaluation of results the Levenshtein distance was used. That is the minimum number of edit operations (insertion, deletion or substitution) needed to be performed to go from one vector to another. For each unlabelled video distance $D(T, L)$ was computed, where T is the truth vector of labels, and L is our predicted vector of labels. This distance is also known as "edit distance". For example, $D([1, 2], [1]) = 1$, $D([1, 2, 3], [2, 4]) = 2$, $D([1, 2, 3], [3, 2]) = 2$.

The overall score for a batch was computed as a sum of Levenshtein distances divided by the total number of gestures performed in the batch. This is similar to an error rate (but can exceed 1). We multiply the result by factor of 100 to resemble the fail percentage. For simplicity, in the rest of this work, we call it error rate.

Chapter 3

Preprocessing

In this Section we discuss possible problems in the dataset as well as the solutions. In the first subsection we focus on depth noise removal. Later we describe the need for trimming the videos — removing set of frames — and the method employed.

3.1 Depth noise removal

One of the problems present is the noise (or missing values) in the depth data. Whenever the Kinect sensor does not receive response from a particular point, the sensor outputs a 0, resulting in black areas visualized in Figure 3.1. This noise occurs usually along edges of objects or, particularly in this dataset, humans. We can see the noise also if the object is out of range of the sensor — 0.8 to 3.5 meters.

Figure 3.1: Examples of depth images with various levels of noise

The level of noise is usually the same within a single batch. However, there is a big difference in level of noise across different batches. If the level is not too high, it looks like 'salt and pepper' noise. Later, in Section 4, we use Histograms of Oriented Gradients (HOGs), which work best with sharp edges, so we need a filter that preserves the edges. One of the best filters for removing this kind of noise is the median filter, and also has our desired

property. Median filter replaces every pixel with median of pixels in small area around itself. The effect of median filter is shown in Figure 3.2. We can see this filter cannot deal with too big areas of noise, however, this is not a problem in our methods. As mentioned earlier, HOG features are sensitive to the edges, but these large areas usually occur along the edges, so the difference in computed features will not be significant.

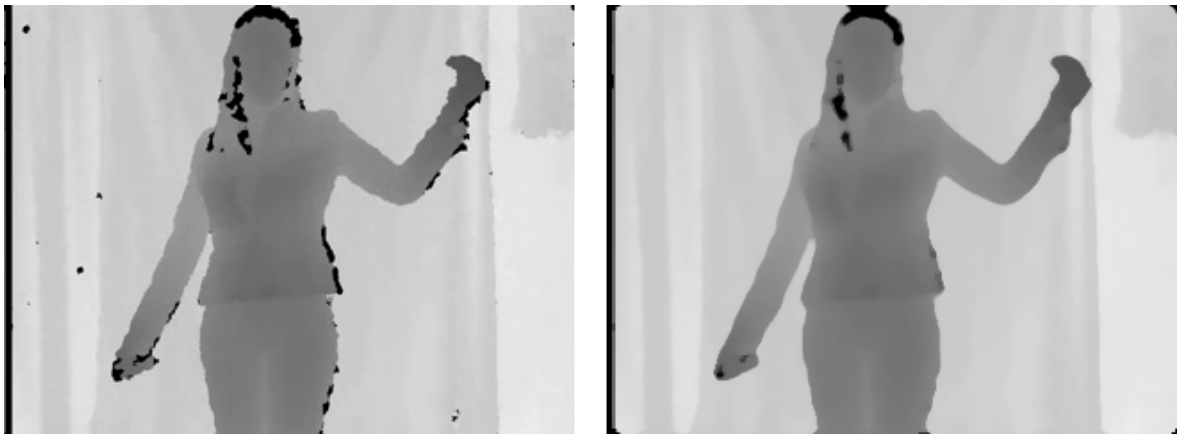

Figure 3.2: Effect of median filter on depth image

3.2 Trimming

In most batches we can find videos with quite long parts at the beginning or at the end of the video, where nothing important happens. Sometimes the user is not moving at all, sometimes trying to turn on/off the recorder.[1] Another problem occurring less often is in batches, where gestures are rather static. There is often variation in time the user stays in particular gesture setting.[2] This is a problem for most possible approaches for tackling the one-shot-learning problem. A solution can be to remove frames from the beginnings and ends of the videos, as well as a part with too big inactivity from middle parts.

One possible approach to removing parts of inactivity can be to watch the amount of motion in the video, and remove parts where nothing happens. This is the idea we employed.

A naive but effective way is to take the depth video and compute differences for every pixel between two consecutive frames. Taking depth videos allows us to ignore problems of texture of clothing or background. We then count simply the number of pixels whose change exceeds a given threshold, or we can simply sum the differences. After numerous

[1] An example is batch devel12, video 23.

[2] An example is batch devel39, particularly video 18.

experiments we ended up with Algorithm 1 described in detail in Appendix A. Suppose we have a video, n frames long. First we remove the background[3] from individual frames and apply the median filter. Then we do not compute differences of consecutive frames, but rather between frames i and $i + 3$. This is to make the motion curve smoother and thus the method more robust. We also found important to even out the amount of motion between, for instance, hand in front of body and hand in front of background. To that end, we set an upper boundary constraint on the difference at 15 (on a scale 0 to 255). Then we computed the actual motion as an average of differences between the chosen frames, as previously described, *above* particular frame, for example

$$\begin{aligned} motion(2) &\leftarrow (mot(1) + mot(2))/2, \\ motion(12) &\leftarrow (mot(9) + mot(10) + mot(11) + mot(12))/4. \end{aligned} \tag{3.1}$$

In *mot* variable we store the average change across all pixels. Then we scaled the motion to range $[0, 1]$.

Once we have the motion in expected range, we can start actually removing frames. At first, we remove sequences from beginning and end of the video with motion below a *threshold* (set to 0.1), under the condition that they are of length at least *minTrim* (set to 5) frames. Then we find all sequences in the middle of the video with motion below the *threshold* of length more than 5, and uniformly choose 5 frames to remain in the video. For example if we were to trim a sequence of length 13, only frames $\{1, 4, 7, 10, 13\}$ would remain. Then we return the video with remaining frames.

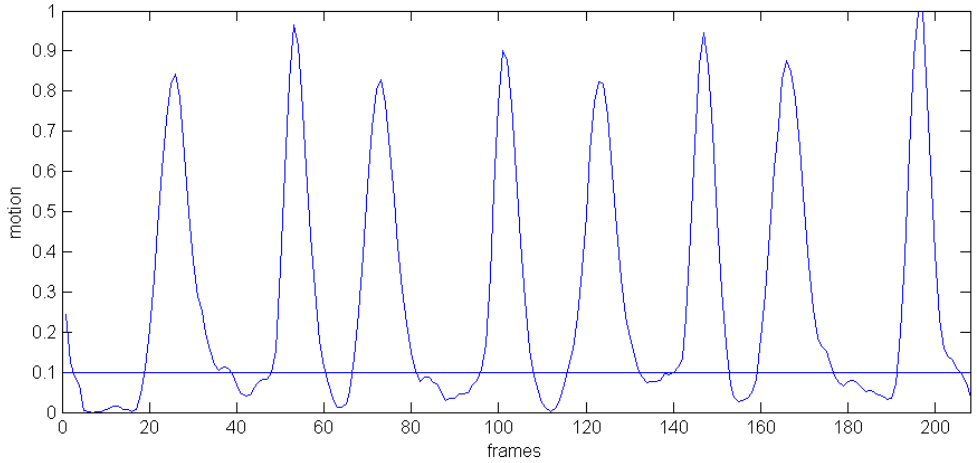

Figure 3.3: Example of a motion graph, batch devel11, video 32

[3]Using an algorithm *bgremove* provided in sample code of the Challenge [3].

One possible modification of this algorithm is in the step in which we scale the motion to the range of $[0, 1]$. In this case, we simply subtract $min(motion)$, and divide by $(max(motion) - min(motion))$. However, especially in videos with 4 or 5 gestures, sometimes a too big outlier makes a problem, because then the threshold is too big. Since the motion curve tends to be relatively smooth, we could instead of $max(motion)$ choose the value of the second highest local maximum. This scaling performs slightly better on long videos, but does not work on short videos. Since we do not know how many gestures to expect in advance, we used the simpler method.

Chapter 4

Feature Representation and Distance Measure

In this Section, we briefly describe the methods used for extracting features we use. Different gestures differ from each other both in appearance and the amount of motion while performing a particular gesture. A good descriptor of the static part of a gesture is the Histogram of Oriented Gradients, proposed by [4]. A good method for capturing size and direction of motion is computing the Optical Flow using the Lucas-Kanade method [11, 17] and creating a histogram of flow. Motivation behind these choices is explained in Section 1. Finally, we describe the Quadratic-Chi distance family proposed by [20] for measuring distances between histograms.

4.1 Histogram of Oriented Gradients

In this section we briefly describe the HOG features. The basic idea is that a local object appearance and shape can often be characterized rather well by the distribution of local intensity gradient (or edge) directions, even without precise knowledge of the corresponding gradient (or edge) positions. In practice this is implemented by dividing the image window into small spatial regions ("cells"), for each cell accumulating a local 1-D histogram of gradient directions (or edge orientations) over the pixels of the cell. It is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy" over somewhat larger spatial regions ("blocks") and using the results to normalize all of the cells in the block.

We used a simple $[-1, 0, 1]$ gradient filter, applied in both directions and discretized the gradient orientations into 16 orientation bins in $0^\circ - 180^\circ$. We had cells of size 40×40 pixels and blocks of size 80×80 pixels, each containing 4 cells. The histogram in each

cell is normalized with sum of euclidean norms of histograms in the whole block. Each cell (except marginal ones) belongs to 4 blocks, thus for one cell we have 4 locally normalized histograms, the sum of which is used as resulting histogram for the cell. Since this method cannot be used to normalize histograms of marginal cells, from 240×320 image we get only 4×6 spatial cells of 16 orientation bins each. Figure 4.1 provides an example visualisation of the HOG features at their actual resolution. The space covered is smaller than the original image, but that is not a problem, since the gestures from the dataset are not performed on margin of the frames. Authors of [4] conclude, that fine-scale gradients, fine orientation binning, relatively coarse spatial cells, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good performance.

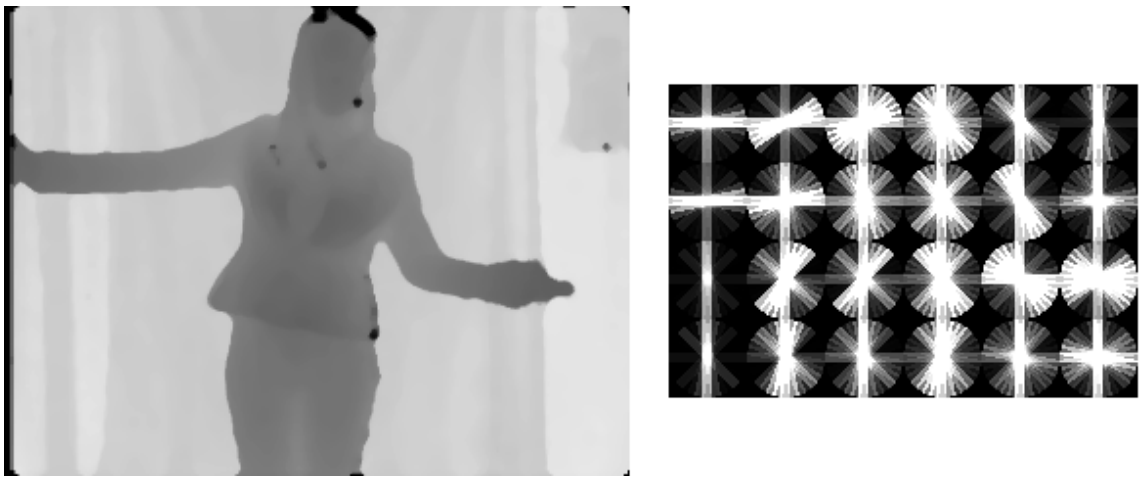

Figure 4.1: Example visualisation of the HOG features

As in Figure 4.1, we computed the HOG features from depth images, since it captures only the edges we are interested in, and not textures of clothing and so on. We used the efficient implementation from Piotr's toolbox [5], function
`hog(image, 40, 16)`.

4.2 Histogram of Optical Flow

In this section we describe the general optical flow principle and the Lucas-Kanade method [11, 17] for estimating the actual flow. The optical flow methods try to estimate the motion between two images (in our case two consecutive frames of video), at times t and $t + \Delta t$ at every position. For our case, a location (x, y, t) with intensity $I(x, y, t)$ will have moved by Δx, Δy, and Δt between the two frames. The following *image constraint equation* can be

given:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t).$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ can be expanded with Taylor series to

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + H.O.T^{1}$$

Taking these two equations, and dividing by Δt we get:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0$$

or

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0,$$

where V_x and V_y are the components of velocity, or optical flow in (x, y, t), and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the partial derivatives of the image at (x, y, t) in corresponding directions.

This is an equation in two variables and cannot be solved without additional constraints. All optical flow methods introduce additional conditions for estimating the actual flow. The Lucas-Kanade method assumes that the flow is essentially constant in a local neighbourhood of the pixel under consideration, and solves the basic optical flow equation for all the pixels in a neighbourhood. The velocity vector $\left(V_x, V_y\right)$ must satisfy

$$\begin{aligned}
&\frac{\partial I}{\partial x}(q_1)V_x + \frac{\partial I}{\partial y}(q_1)V_y + \frac{\partial I}{\partial t}(q_1) = 0,\\
&\frac{\partial I}{\partial x}(q_2)V_x + \frac{\partial I}{\partial y}(q_2)V_y + \frac{\partial I}{\partial t}(q_2) = 0,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
&\frac{\partial I}{\partial x}(q_n)V_x + \frac{\partial I}{\partial y}(q_n)V_y + \frac{\partial I}{\partial t}(q_n) = 0,
\end{aligned}$$

where $q_1, q_2, \ldots, q_n$ are the pixels inside the neighbourhood and the partial derivatives are evaluated at points q_i at the current time t. The equations can be written in matrix form $Av = b$, where

$$A = \begin{pmatrix} \frac{\partial I}{\partial x}(q_1) & \frac{\partial I}{\partial y}(q_1) \\ \frac{\partial I}{\partial x}(q_2) & \frac{\partial I}{\partial y}(q_2) \\ \vdots & \vdots \\ \frac{\partial I}{\partial x}(q_n) & \frac{\partial I}{\partial y}(q_n) \end{pmatrix}, \quad v = \begin{pmatrix} V_x \\ V_y \end{pmatrix}, \quad b = \begin{pmatrix} \frac{\partial I}{\partial t}(q_1) \\ \frac{\partial I}{\partial t}(q_2) \\ \vdots \\ \frac{\partial I}{\partial t}(q_n) \end{pmatrix}.$$

[1] Higher Order Terms

This system is usually over-determined, since it has more equations than unknowns. The Lucas-Kanade method obtains a solution by the least squares principle. Thus the solution is

$$v = \left(A^T A\right)^{-1} A^T b.$$

After obtaining the optical flow in every point of the image we divide the image (of 240×320 pixels) to a grid of 6×8 spatial cells. We then put each optical flow vector into one of 16 orientation bins in each spatial cell, and scale them so they sum to 1 to get a histogram of $6 \times 8 \times 16$ fields. We also tried to scale in each spatial cell separately, and the difference of error rate in our methods on all development batches was less than 0.5. We computed the optical flow from color videos, converted to grayscale, again using efficient implementation of the Flow estimation from Piotr's toolbox [5], function

```
optFlowLk(image1, image2, [] , 4, 2, 9e-5);
```

4.3 Measuring Distance of the Histograms

Our method relies on making comparisons between pairs of frames in two videos, which requires as a component, to measure differences between histograms. The relatively simple methods based on the sum of bin-to-bin distances suffer from the following limitation: If the number of bins is too small, the measure is not discriminative and if it is too large it is not robust. Distances, that take into account cross-bin relationships, can be both robust and discriminative. With the HOG and HOF feature at the resolution that we selected, simple bib-to-bin comparisons are not robust, as exemplified in Figure 4.2. Thus we would like a measure that would look into surrounding orientation bins and, after experimenting, also to surrounding spatial cells. Thus we would also like a measure, that would reduce the effect of big differences, and also look into surrounding spatial cells. We adopted the following Quadratic-Chi distance family introduced by [20].

Let P and Q be two histograms. Let A be a non-negative symmetric bounded bin-similarity matrix, such that each diagonal element is bigger or equal to every other element in its row. Let $0 \leq m < 1$ be a normalization factor. A Quadratic-Chi histogram distance is defined as:

$$QC_m^A(P,Q) = \sqrt{\sum_{i,j} \left(\frac{(P_i - Q_i)}{\left(\sum_c (P_c + Q_c) A_{ci}\right)^m} \right) \left(\frac{\left(P_j - Q_j\right)}{\left(\sum_c (P_c + Q_c) A_{cj}\right)^m} \right) A_{ij}}$$

where we define $\frac{0}{0} = 0$. The normalization factor m reduces the effect of big differences (the bigger it is, the bigger reduction; in our methods set to 0.5). During comparing i^{th} orientation bins of two histograms, we want to look into the matching orientation bins, to

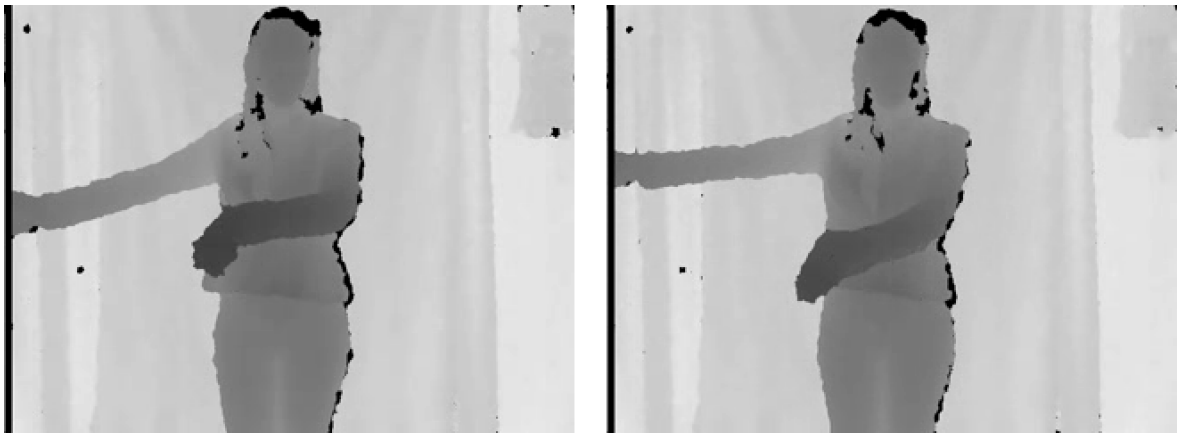

Figure 4.2: Example of need for cross-bin similarities: the same moment in performance of the same gesture in two different videos. The right hand stays at the same place, the left hand is moving. This illustrates how the same element can result in different neighbouring orientation bins in HOG being big in different cases.

4 surrounding orientation bins (2 left, 2 right), and into the same orientation bins within 8 surrounding spatial cells. MATLAB code for creating the matrix A which captures these properties is in Appendix B.

Chapter 5

Recognition

In this Section we describe the methods used for recognition. In our first method we create a single model and look for cheapest path of a new video through the model. In our second method we create a separate model for every training video and using sliding frame window look for similar parts of training videos.

5.1 Single Model — Dynamic Time Warping

In this method (we will call it SM) we use both Histograms of Oriented Gradients and Histograms of Optical Flow and perform temporal segmentation simultaneously with recognition.

At first, we create a model illustrated in Figure 5.1 for whole batch. Every row in the Figure represents a single training video. Every node represents single frame of the video. In a node we store HOG and HOF features belonging to the particular frame. Recall that the HOF needs two consecutive frames. Thus if a video has f frames, the representation of this video has $f - 1$ nodes, ignoring the HOG of first frame. We add an arbitrary node, called Resting Position (RP), obtained as average representation of first frames of each video.

We want to capture the variation in speed of performing gestures, thus we set the transitions in the following way: When being in a particular node n in time t, moving to time $t + 1$ we can either stay in the same node (slower performance), move to node $n + 1$ (the same speed of performance), or move to node $n + 2$ (faster performance). Experiments suggested allowing transition to node $n + 3$ is not needed with the trimming described in Section 3. It even made the whole method perform worse. From the RP we can move to the first three nodes of any video, and from the last three nodes of every video we can move to the RP.

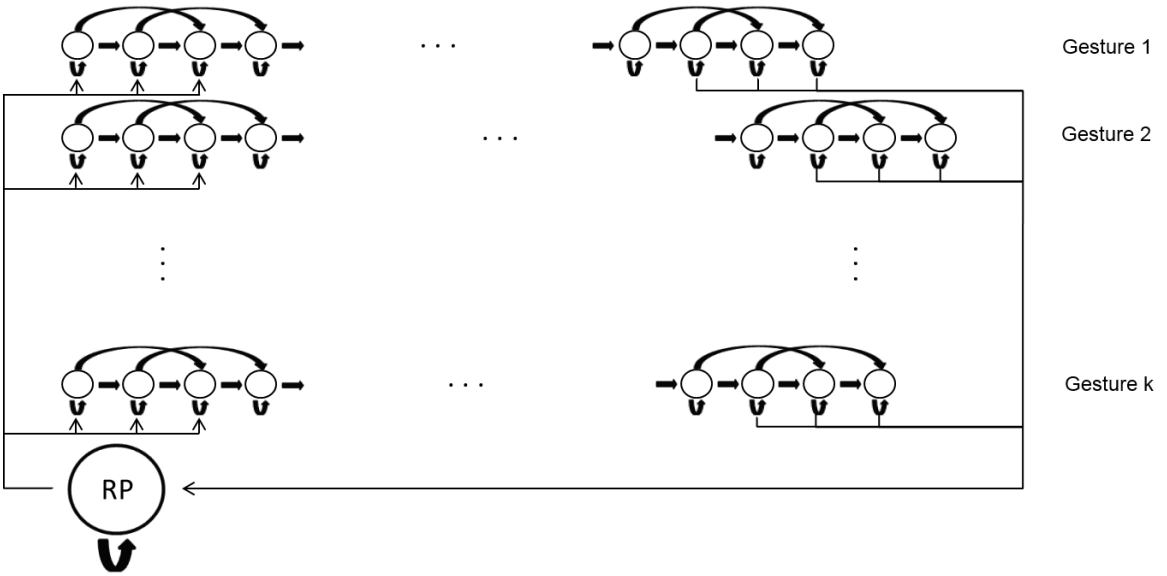

Figure 5.1: Model for Dynamic Time Warping

When we have this model, we can start inferring the gestures present in a new video. First, we compute representations of all the frames in the new video. Then we compute similarities of every node of our model with every frame representation of the new video. We compute similarities of both matching HOGs and HOFs, using the Quadratic-Chi distance described in Section 4.3, and simply sum the distances. This makes sense since the empirical distribution functions of distances of HOGs and HOFs are similar. We can represent these distances as a matrix of size $N \times (F - 1)$, where N is the number of all nodes in the model, and F is the number of frames in the new video. Using the Viterbi algorithm we find the shortest path through this matrix (we constrain the algorithm to begin in RP or in any of first three nodes of any gesture). Every column is considered a time point, and in every time point we are in one state (row of the matrix). Between neighbouring time points the states can change only along the transitions in the model. This approach is also known as Dynamic Time Warping [2].

Result of the Viterbi algorithm is a path — sequence of nodes which correspond to states in which our new video was in time. From this path we can easily infer which gestures were present (which rows in Figure 5.1), and in what order. The flow of states in time is displayed in Figure 5.2 (the color represents the cumulative cost up to a particular point — the darker the bigger).

Figure 5.2: Example of flow of states in model — devel01, video number 11 — true labels are {9, 4, 4, 9}. The gray levels represent the shortest cumulative path ending in a particular point.

5.2 Multiple Models — Sliding Frame

In this method (we will call it MM) we used only the Histogram of Oriented Gradients and perform temporal segmentation prior to recognition. We created similar model as in SM, but separately for every training video, illustrated in Figure 5.3. Again, every node represents HOG features of a single frame. Thus if we have k different gestures, we have k similar models. We do not need a RP node, since we will be looking for short sequences in these models similar to short sequences of frames of a new video. Again, the transitions between states in models want to capture variation in speed of performing gestures.

Figure 5.3: Model for every video in MM

MM differs from *SM* mainly in approach to inferring gestures present. First, we compute all the HOG representations of a new video and compute their similarities with all the nodes in k models. Then we employ the idea of sliding frame. The idea is to take a small sequence of the new video and monitor which parts of which training videos does it resemble to. First we select frames 1 to l (we set $l = 10$) and treat this similarly as in *SM*. We look for the shortest path through our first model without constraint on where to begin or end. We do the same with every model. This results in numbers representing the resemblance of a small part of our new video with any part of every training video, and optionally also numbers of nodes resembling it. Then we select frames 2 to $(l + 1)$, repeat the whole process, and move forward through the whole video.

Finally we obtain a matrix of size $k \times (f - l + 1)$, where k is the number of gestures and f number of frames in new video. Every column represents a time instant and every row a gesture. An example of such matrix is shown in Figure 5.4. Humans can fairly easily learn to recognize where and which gestures are present, but it is a bit more challenging task for a computer. We tried to treat columns as feature vectors and feed it to *SM* and tried to build a Hidden Markov Model to infer gestures present. We also tried to include information of what nodes of a particular model were present for every time instant, so we can prefer gestures where most of the nodes were included. That was difficult to take into account, because the start and end of most videos are very similar (Resting Position). All the methods had problems identifying two identical gestures occurring after each other, and also two similar gestures occurring after each other. We did not find satisfactory solutions to these problems without deteriorating performance.

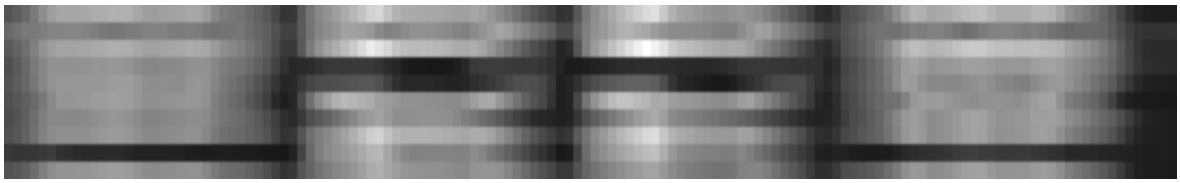

Figure 5.4: Example of sliding frame matrix — devel01, video number 11.

None of these methods manages to beat the naive approach. We resorted to first segment the video using an algorithm provided by organizers in the sample code called *dtw_segment*. The algorithm is very fast and segments the videos very well. After segmenting, we simply summed along the rows in corresponding parts of the scores matrix and picked the minimum. An improvement was to perform a weighed sum that emphasizes the center of the video, since the important information is usually in the middle.

We used only HOG features in this method because every try to include HOF features gave considerably worse results. One possible explanation for this can be we do not need to focus on the overall movement while looking only for short segments of videos, but it is more important to capture the static element.

Chapter 6

Results

In this section we discuss results of our methods. We also compare our results with those of other challenge participants as well as with other already published methods with experiments on this dataset.

All our experiments were conducted on a processor Intel Core i7 3610QM, with memory 2×4GB DDR3 1600 MHz. The running time of SM was approximately 115% of real-time (takes longer to process than to record), while MM was approximately 90% of real-time. However, none of our methods could be trivially converted to an online method, since we need to have the whole video in advance.

We summarize the performances of our methods on all available datasets in Table 6.1. The results also show our preprocessing steps positively influences the final results. The MM looks better on the first 20 development batches, but performs worse overall. All other published works provide results only on first 20 batches, which is too few for any reliable conclusions. Therefore we suggest providing results on all the batches for bigger relevance.

As mentioned in Section 1, we chose our descriptors to exploit specific properties of the dataset — the user stays at the same place, and thus the important parts of gestures have always roughly the same position within the image. Hence it is not surprising that our model is not translation nor scale invariant. Organizers of the challenge [7] created 20 translated and scaled data batches, and analyse robustness of methods of top ranking participants. In general, the bag-of-features models have this property, but they are usually rather slow. If we wanted to incorporate translation invariance, one method could be to extract body parts from image (the algorithm is provided within Kinect Development Toolkit) and align the images so the user is at the same position.

Batches	SM	MM
devel01-20	23.78	21.99
devel01-480	29.40	34.43
valid01-20	20.01	24.48
final01-20	17.02	23.08
final21-40	10.98	18.47
devel01-20 (without trimming)	26.24	22.82
devel01-20 (without medfilt)	24.70	23.92
devel01-20 (SM; only HOG)	24.53	
devel01-20 (MM; HOG and HOF)	28.73	

Table 6.1: Overview of our results on datasets

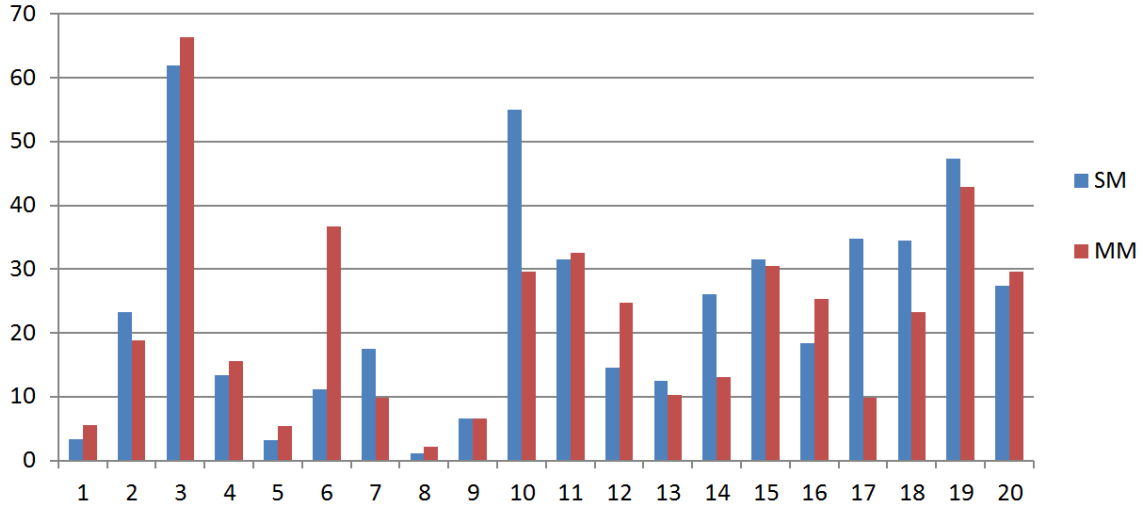

Figure 6.1: Scores of our methods on first 20 development batches

The results of our method on each of the first 20 batches is displayed in Figure 6.1. Often our methods perform similarly, but one can spot significant differences in batches devel06 (SM — 11.11, MM — 36.67), devel10 (SM — 54.95, MM — 29.67), devel17 (SM — 34.78, MM — 9.78). In batches devel10 and devel17, the gestures are only static and all occurs in the same place in space. In this particular setting, the information about any motion (HOF) can be redundant. This can be a reason why MM performs better, since we do not include any motion descriptors into the representation. In devel06, the problem is, the gestures are performed very quickly, thus the videos are often very short. This is a problem since the matrix in Figure 5.4 has only few columns, resulting in poor performance of MM.

But this brings us to a new preprocessing step. Suppose we have more algorithms for solving this one-shot-learning task. If we were able to describe in any way what types of gestures does which algorithm recognize the best, we could boost the overall performance by picking the right algorithm in advance, after seeing the training videos. This is a problem we have unsuccessfully tried to solve, and which remains open for future work. If we always pick the better from our two methods, we would achieve score of 19.04 on the batches devel01-20.

The methods used by other challenge participants — alfnie, Pennect, Joewan [21], OneMillionMonkeys, Manavender [19] — are summarized by [8, 7]. We briefly describe other published works applied on this dataset. We provide comparison of all of these methods in Table 6.2.

Method / team	devel01-20	valid01-20	final01-20	final21-40
SM (ours)	23.78	20.01	17.02	10.98
MM (ours)	21.99	24.48	23.08	18.47
alfnie	NA	9.51	7.34	7.10
Pennect	NA	17.97	16.52	12.31
Joewan	19.45	16.69	16.80	14.48
OneMillionMonkeys	NA	26.97	16.85	18.19
Mananender	26.34	23.32	21.64	19.25
Wu et al.	26.00	25.43	18.46	18.53
BoVDW	26.62	NA	NA	NA
Lui	28.73	NA	NA	NA
Fanello et al.	25.11	NA	NA	NA

Table 6.2: Comparison of results of methods from the competition as well as published methods

Wu et al. [23] pre-segment videos and represent motions of users by Extended-Motion-History-Image and use maximum correlation coefficient classifier. The Multi-view Spectral Embedding algorithm is used to fuse duo modalities in a physically meaningful manner.

The paper [9] present a Bag-of-Visual-and-Depth-Words (BoVDW) model for gesture recognition, that benefits from the multimodal fusion of visual and depth features. They combine HOG and HOF features with a new proposed depth descriptor.

Tensor representation of action videos is proposed by [18]. Aim of his work is to demonstrate the importance of the intrinsic geometry of tensor space which yields a very discriminating structure for action recognition. The methods is assessed using three gesture databases, including Chalearn gesture challenge dataset.

Finally Fanello et al. [6] develop a real-time learning and recognition system for RGB-D images. The proposed method relies on descriptors based on 3D Histogram of Flow, Global Histogram of Oriented Gradient and adaptive sparse coding. The effectiveness of sparse coding techniques to represent 3D actions is highlighted in their work.

Chapter 7

Discussion and Conclusions

In this thesis we presented two methods for solving the one-shot-learning gesture recognition task introduced in the *ChaLearn Gesture Challenge* [3]. We have significantly helped narrow down the gap between human and machine performance (the baseline method achieved 50% error rate on final evaluation set, our method 11%, while the human error rate is under 2%). Our methods outperform other published methods and we suggest that other authors provide results on whole dataset for greater relevance of achieved results.

We combine static — Histograms of Oriented Gradients — and dynamic — Histogram of Optical Flow — descriptors in first method, where we create one model and perform temporal segmentation simultaneously with recognition using Dynamic Time Warping. We use only static descriptors and use pre-segmentation as a preprocessing step in second method, where we look for similar parts in training videos using sliding frame.

Our first method is similar to one developed by team Pennect in the Challenge, and also performs similarly. They also used HOG features, but at different scales, and used a one-vs-all linear classifier, while we use the Quadratic-Chi distance [20] to measure distances between individual frames. The recognition was also parallel with temporal segmentation using a DTW model. Surprisingly, the Pennect team used only the color images.

Bag-of-features models provide comparable [21] of slightly worse [9] results than ours. The advantage against our methods is they are scale and translation invariant - which is necessary for real-world applications like in gaming industry. On the other hand, these methods rely on presegmentation of videos to single gestures, and are considerably slower, hence currently not applicable. An interesting property of these methods is their results seem to have lower variance — error rate at difficult datasets (for instance devel10) is smaller, but struggle to obtain strong recognition rate on easy datasets (devel08, devel09).

We present a novel video trimming technique, based on amount of motion. Its motivation is to remove unimportant segments of videos and thus reduce probability of confusing gestures. The method improves overall results of our methods (Table 6.1), and small improvement was confirmed by [23] — 2% and [21] — 0.5%.

Finally, we suggest an area for future work. Having more well working methods at disposal, we can analyse their results on different types of gesture vocabularies, users and other settings. Overall performance could be boosted if we were able to decide which recognizer to use in advance. Especially, deeper analysis of differences of results between Bag-of-words models and Dynamic Time Warping models is needed to obtain better description of their behaviour on different types of gesture recognition tasks.

Appendix A

Algorithm 1

In this Appendix, we provide algorithm for trimming video described in Section 3.2.

Algorithm 1 Trimming a video

$n \leftarrow length(video)$
$gap \leftarrow 3 \quad maxDiff \leftarrow 15 \quad threshold \leftarrow 0.1 \quad minTrim \leftarrow 5$
for $i = 1 \rightarrow n$ **do**
    $video(i) \leftarrow bgremove(video(i))$ ▷ Background removal
    $video(i) \leftarrow medfilt(video(i))$ ▷ Median filter
end for
for $i = 1 \rightarrow (n - gap)$ **do**
    $diff(i) \leftarrow abs(video(i) - video(i + gap))$
    $diff(i) \leftarrow min\{diff(i), maxDiff\}$
    $mot(i) \leftarrow mean(diff(i))$ ▷ Mean across all pixels
end for
$motion \leftarrow avgMotion(mot)$ ▷ As in Equation 3.1
$motion \leftarrow scale(motion)$ ▷ Scale motion so its range is 0 to 1
$frames \leftarrow vector(1 : n)$
if $|beginSequence(motion < threshold)| \geq minTrim$ **then**
    $frames \leftarrow trimBegin(frames)$ ▷ Remove all frames
end if
if $|endSequence(motion < threshold)| \geq minTrim$ **then**
    $frames \leftarrow trimEnd(frames)$ ▷ Remove all frames
end if
for all $|sequence(motion < threshold)| > minTrim$ **do**
    $frames \leftarrow trimMiddle(sequence, frames)$ ▷ Remove all frames but $minTrim$
end for
return $video(frames)$

Appendix B

Algorithm 2

In this Appendix, we provide MATLAB algorithm for creating similarity matrix used in the Quadratic-Chi distance described in Section 4.3. We have histograms of $h \times w$ spatial cells, and p orientation bins in each of the spatial bins.

Algorithm 2 MATLAB code producing the similarity matrix

```
gauss = fspecial('gaussian', 3, 0.56);
B = diag(ones(1,h)) + 2*(diag(ones(1, h-1), 1) + diag(ones(1, h-1), -1));
C = diag(ones(1,w)) + 2*(diag(ones(1, w-1), 1) + diag(ones(1, w-1), -1));
D = kron(C, B); % Kronecker tensor product
D(D == 1) = gauss(5);
D(D == 2) = gauss(2);
D(D == 4) = gauss(1);
A = imfilter( eye(p), gauss, 'circular');
A = sparse(kron(D, A)); % The final similarity matrix
```

Bibliography

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370, 1994.

[3] ChaLearn Gesture Dataset (CGD2011), ChaLearn, California. `http://gesture.chalearn.org/data`, 2011.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005*, volume 1, pages 886–893. IEEE, 2005.

[5] Piotr Dollár. Piotr's Image and Video Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

[6] Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone. One-shot learning for real-time action recognition. 2013.

[7] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Hugo Jair Escalante, and Ben Hamner. Results and analysis of the chalearn gesture challenge 2012. 2013.

[8] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, Ben Hamner, and Hugo Jair Escalante. Chalearn gesture challenge: Design and first results. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–6. IEEE, 2012.

[9] Antonio Hernández-Vela, Miguel Ángel Bautista, Xavier Perez-Sala, Victor Ponce, Xavier Baró, Oriol Pujol, Cecilio Angulo, and Sergio Escalera. BoVDW: Bag-of-Visual-and-Depth-Words for gesture recognition. In *International Conference on Pattern Recognition*, pages 449–452, 2012.

[10] Nazlı Ikizler and David Forsyth. Searching video for complex activities with finite state models. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[11] Takeo Kanade and Bruce D. Lucas. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.

[12] Alexander Klaser and Marcin Marszalek. A spatio-temporal descriptor based on 3d-gradients. 2008.

[13] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[14] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[15] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine learning: ECML-98*, pages 4–15. Springer, 1998.

[16] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[17] Bruce D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

[18] Yui Man Lui. Human gesture recognition on product manifolds. *Journal of Machine Learning Research*, 13:3297–3321, 2012.

[19] Manavender R Malgireddy, Ifeoma Inwogu, and Venu Govindaraju. A temporal bayesian model for classifying, detecting and localizing activities in video sequences. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 43–48. IEEE, 2012.

[20] Ofir Pele and Michael Werman. The quadratic-chi histogram distance family. *Computer Vision–ECCV 2010*, pages 749–762, 2010.

[21] Jun Wan, Qiuqi Ruan, Wei Li, and Shuang Deng. One-shot learning gesture recognition from rgb-d data using bag of features. *submitted to JMLR*, 2013.

[22] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, Cordelia Schmid, et al. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, 2009.

[23] Di Wu, Fan Zhu, and Ling Shao. One shot learning gesture recognition from rgbd images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 7–12. IEEE, 2012.

[24] Tian Xia, Dacheng Tao, Tao Mei, and Yongdong Zhang. Multiview spectral embedding. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(6):1438–1446, 2010.