Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
Graphics, Vision and Robotics

Frame-wise representations of depth videos for action recognition

Gurkirt Singh

June 20th 2013

Research project performed at INRIA Grenoble

Under the supervision of:
Prof. Radu Horaud, Perception Team
Dr. Georgios Evangelidis, Perception Team

Defended before a jury composed of:
Prof. Remi Ronfard
Prof. James Crowley
Prof. Marie-Christine Fauvet
Dr. Zaid Harchaoui
Dr. Georgios Evangelidis

June 2013

Abstract

We investigate the of problem continuous action recognition from depth images. Further, we investigate the frame-wise classification as a solution for the continuous action recognition problem. We present three types of depth data representation from depth frames, which are referred as *single-reference representation*, *multiple-reference representation* and *no-reference representation*. The first two regard a global translation transformation while the third is a local (region-wise) similarity transformation of the depth data. These representations directly provide the descriptors of each depth image. We represent each frame of the video by bag-of-words (BOW) representation. We show that variation of the visual words over a temporal window accounts for the motion information which is integrated in each frame BOW representation, which we encode by temporal smoothing and temporal differentiation of frame BOW representation. Finally, once we have frame representations, standard classifiers such as random forests or support vector machines are used for frame-wise action classification. We apply a simple voting scheme after frame-wise classification for isolated action recognition to compare our method with state-of-the-art methods on isolated recognition and we able to achieve better performance than state-of-the-art methods.

Acknowledgements

First of all I would like express my deepest appreciation to my supervisor Prof. Radu Horaud, who offered me to work on this interesting subject and patiently helped me with any problems and questions that arose during the work on the thesis. I would also like to thanks to Dr. Georgios Evangelidis for his endless help and support during my internship and reviewing manuscript in spite of his busy schedule. He made many important suggestions and helped me to enhance this thesis.

I would also like to thank the members of perception team. Who made my stay pleasant and helped me with many tricks in implementation. Especially, I would like to thanks Kaustubh for his inputs into my work.

I also wanted to express my gratitude to my parents for all their encouragement and support in my studies.

Contents

Chapter 1

Introduction

In recent years, human activity recognition has become a very active research field in the computer vision area due to its involvement in a variety of real world applications, such as visual surveillance, behavior analysis, human computer interaction, content based video search, video indexing and elderly monitoring [9]. Despite the many advances in the related literature, human action recognition remains a challenging problem, due to the large variation of human pose, appearance and intra-class variance of actions, i.e. the same action can be performed in several ways.

Color information was initially used for the action recognition problem [20] [21] [14]. The availability of depth cameras such as kinect at consumer price led to a new class of recognition algorithms that deal with depth data only [29] [42] or combine color and depth information [47]. As opposed to color data, the depth information of a scene is texture invariant and describes its geometric properties that can be exploited to simplify modules such as the foreground segmentation, the human pose description etc. A categorization of the action recognition algorithms with respect to various parameters has been done in [43].

The majority of the works in the related literature, especially those that deal with depth data, assumes that a video contains a single action or the action boundaries are a priori known when a cascade of actions is performed in the video shot. We refer to the classification of such videos as *isolated action recognition*. The case of multiple actions becomes very challenging when no prior information about the boundaries is given, thus the segmentation of the video into single actions and the classification of each one must be done simultaneously. We refer to this solution as *continuous action recognition*.

Obviously, the frame-wise classification provides a solution for the continuous action recognition problem. However, it has been shown that single frames are not very informative and a couple of frames is required to discriminate actions and compete algorithms that use the whole set of frames in a video [33]. Moreover, common descriptors that encode the local information of frames usually describe data captures from successive frames [21] so that the motion information is somehow included.

We investigate here the case where the descriptors count on depth data of single frames. We focus on the frame-wise classification and we investigate the loss in performance compared to the video-wise classification. As opposed to standard methods [20],[21], [14], the motion information is late encoded, that is, after describing the frames through a bag-of-words (BOW) scheme [13]. In other words, it is the variation over time of the quantized data that provides this kind of information. We investigate three types of depth data representation, which are referred here as *single-reference representation*, *multiple-reference representation* and *no-reference rep-*

resentation that are discussed in detail in Chapter 4. The first two regard a global translation transformation while the third is a local (region-wise) similarity transformation of the depth data. Note that for the single-reference representation color data are also required to extract a reference point for each frame. A BOW scheme applies in the resulting descriptors which allows for video- of frame-wise vector representations. In the latter case, the variation of the visual words over a temporal window accounts for the motion information which is integrated in each frame BOW representation; this is also discussed in detail in Chapter 5. Finally, once we have frame or video representations, standard classifiers such as random forests [6] or support vector machines [10] are tested and compared. Figure 1.1 shows a diagram of our pipeline. To the best of our knowledge, frame-wise classification of depth video sequences has not been done before. It is also important to note that the investigation of a frame-wise classification is worthwhile since any global optimization scheme for the continuous action recognition problem counts on the discrimination between single frames or short-time sequences [17]. As an example, a two-pass dynamic programming algorithm can easily apply once each frame is roughly classified [19].

The state-of-the-art action recognition algorithms that mostly rely on depth data are discussed in the next chapter. Most of the algorithms follow a standard pipeline that consists in defining a classifier and appropriately constructing its input; the commonly used input results from a BOW-based data mapping. The general aspects of these two steps are described in Chapter 3. In Chapter 6, we evaluate the performance of our schemes with respect to different parameters using publicly available datasets. Furthermore, we compare our algorithms with state-of-the-art isolated action recognition methods in chapter 6. Finally, the conclusions of this work are drawn in the last chapter.

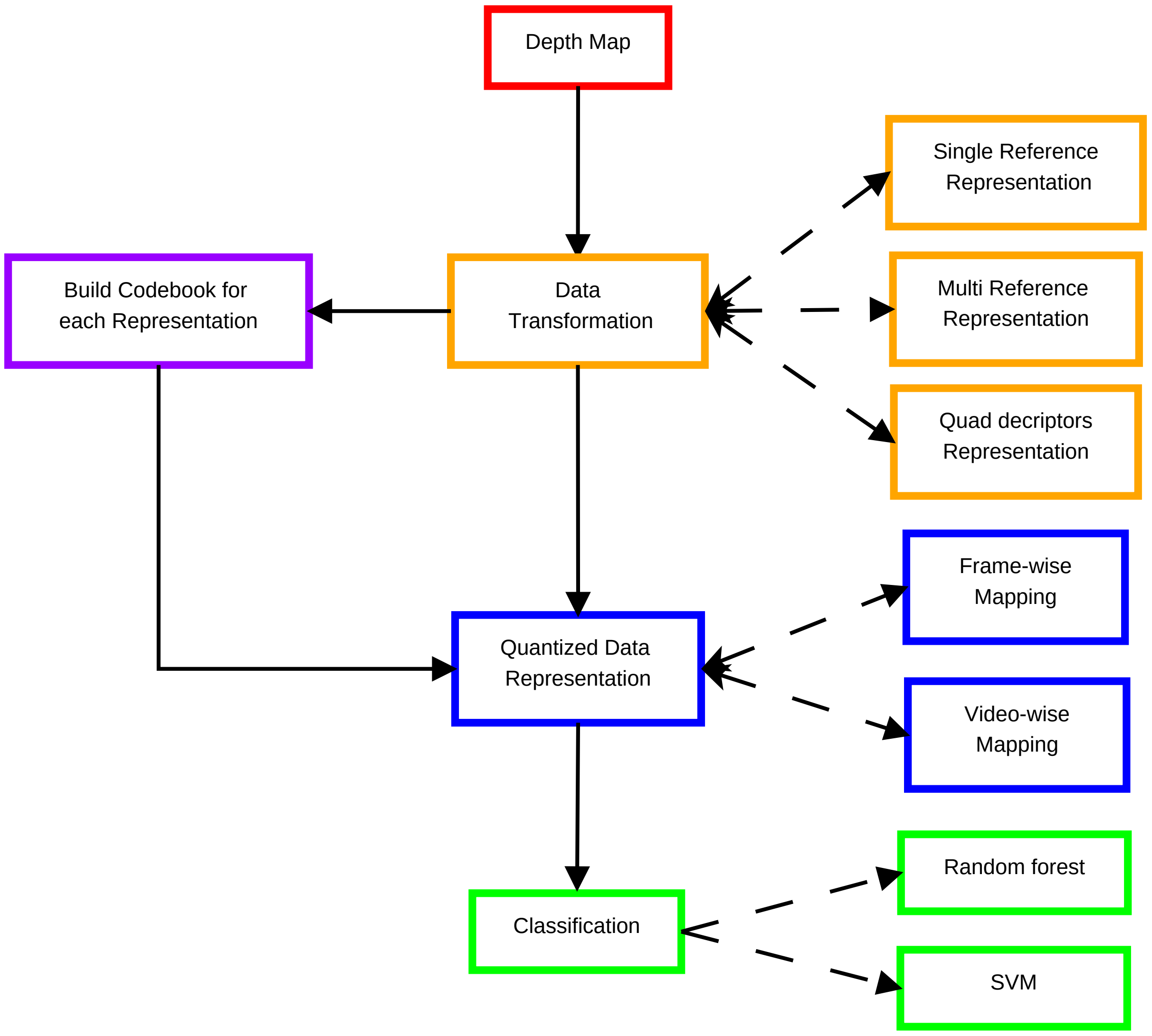
Depth Map
Data Transformation
Build Codebook for each Representation
Single Reference Representation
Multi Reference Representation
Quad decriptors Representation
Quantized Data Representation
Frame-wise Mapping
Video-wise Mapping
Classification
Random forest
SVM

Figure 1.1: Overview of the system

Chapter 2

State-of-the-art

An articulated model of human body could give reliable features to perform action recognition. But to obtain such a model from intensity images and depth images not a easy task. Much of the action recognition research using depth cameras is based on the use of human skeleton joints information provided by the algorithm in [36]. They use simple depth comparison feature around each pixel over set of offset values. Combining with random forest, it presents one of the greatest advances in human body pose estimation and expressing it as a set of joints using depth image. Use of skeleton joints and tracking over time has been exploited for human action recognition in [41], [42], [46], [45] etc. Wang *et al*. [41] uses 3D joint positions and define a local occupancy pattern around each joint based on the depth appearance. In their later work [42], they argue that pairwise relative positions of joints are more discriminative features. Also, the local appearance around joints need to be taken into account to capture the interaction with other objects. On the other hand, Xia *et al*.[45] uses histogram of 3D joints location, that characterizes human postures. Yang *et al*. [46] propose a method to compute features from joints positions, which contain the information of pose and motion. feature is based on the position differences of joints. They claims that first 15-20 frames are sufficient on MSR Action3D dataset [23] to perform action recognition comparable to the one obtained from whole action video.

Skeleton data is always not reliable especially in case of occlusion. It can have errors, when person is not directly facing the cameras, which limits the performance of the action recognition systems based on the skeleton joints. To overcome this problem [29] describe the action sequence in $4D$ space (space, depth, time) using the surface normals to the 4D surface. They describe the action by quantization of the surface normals in to a histogram. They propose method for non-uniform quantization of the histogram bins to make them more discriminative. Their description of the sequence preserve the order in frame, to capture the temporal information in video. But, their algorithm requires the background removal to extract the 4D surface normals.

Bag of 3D points has shown promises in [23]. They try to characterize a set of salient postures and employ an action graph [24] to model the action dynamics. But,the algorithm is tested on clean dataset. They use the segmented human body and three orthogonal Cartesian planes are used to project 3D points. They sample the points from the contours of the projections planes. Novelty of their system is that they only use 1% 3D of all points on human body.

Most of the method available either requires skeleton information or human body segmentation, which may not be available all the time in practice. Other solutions are needed to avoid this problem. Spatial-temporal interest point feature are widely applied in action recognition in intensity images [21], [14], [20] etc. Following the work in intensity images on local spatio-temporal interest points (STIPs), Xia *et al*. [44] proposed a filtering method to extract STIPs from depth images. On the top of these extracted point they build a depth cuboid similarity feature.

Our method does not require skeleton or segmentation of human body. We propose three frame representations, which intrinsically encode the pose information in a particular frame. Each frame is represented by bag-of-word scheme and we observe that undesirable background descriptors appear in every frame of the training data, due to this the inverse document frequency for background words is zero. So background descriptors are removed.

Chapter 3

Parts of action recognition system

Action recognition is commonly divided into three parts. 1) feature extraction, 2) bag-of-words (BOW) representation and 3) classification. The first part regards the extraction of appropriate features from video which is usually local or global descriptors of color and/or depth data. This part for our algorithm is described in detail in next chapter. This chapter discusses the BOW approach in the context of standard image classification framework as well as two commonly used classification methods.

3.1 BOW representation

Once Cruska *et al*. [13] proposed bags of keypoints for visual categorization; since then, it has been successfully applied to many computer vision problems. Sivic *et al*. in [38] gave new direction to the field of content-based image retrieval, video retrieval and image classification. The BOW approach is used for efficient image retrieval while it provides a compact compact representation of each image i.e., each image is described by a histogram vector of equal length. The steps involved in computing BOW representation are as follow.

- Feature extraction and descriptor computation
- Image representation using vector quantization
- Histogram weighting

3.1.1 Feature Extraction

Images features are usually extracted from local regions using interest point detectors, such as Harris corner [16], scale and affine invariant interest point detectors [26] and space time interest points [14], [21] etc. Descriptors encode the data around each interest point. Depending on the application in question, the descriptors are computed from appearance histograms of oriented gradients(HOG), or motion histograms of optical flow (HOF) or combination of appearance and motion descriptors [40].

3.1.2 Image representation using vector quantization

Descriptors are gathered from whole training data and are put together in a pool. Vector quantization is done by using unsupervised learning clustering algorithm such as K-means. K-means applies on this pool and clusters (quantizes) all the descriptors, let $(d_1,...d_i....d_n)$ with $d_i \in R^N$ into K clusters. K-means is a NP -hard problem, but there is a heuristic algorithm [25] that minimizes the within clusters sum of squares of distances, which converges to a local optimum solution. After multiple initializations of cluster centers, K-means give a reasonably good quantization of the descriptors pool. It partitions the n descriptors into K clusters $L = \{L_1,...L_k...L_K\}$, by solving the following minimization problem

$$L = \underset{L\prime}{\text{argmin}} \sum_{k=1}^{K} \sum_{d \in L'_k} \|d - \mu\left(L'_k\right)\|^2$$

where $\mu\left(L'_k\right) \in R^N$ is the cluster center of L'_k.

The above process gives a codebook or vocabulary parametrized by the cluster centers. The only parameter that has to be tuned is K, which denotes the number of words in the visual vocabulary. K should be big enough to capture the inter-class variance and small at the same time to keep the representation compact. Now, given the image descriptors, each image can be described as a histogram of visual words, where each bin is the index of each visual word.

3.1.3 Histogram weighting

Tf-Idf weighting is helpful to downweight the non discriminative words. In essence it downweights the words that appear in too many images. It is shown in experiments that words corresponding to background are removed by using the weighting scheme. The Tf-Idf weighting score for each histogram element is defined by the product of two scalars which are defined below, the term frequency and inverse document frequency.

Weighting with tf-idf score: Visual words are weighted on their frequency.
Term frequency (Tf): Normalized frequency of the i^{th} term (word) t_i in their j^{th} document (image) (I_j).

$$tf_{ij} = n_{ij} / \sum_{k=1}^{K} n_{kj}, \qquad i = 1,2,....K, j = 1,2,.....J$$

where n_{ij} is frequency of word t_i in image I_j

Inverted Document frequency Idf: Total number of documents (images) divided by number of documents (images) containing the term t_i

$$idf_i = log\frac{|J|}{|J_{t_i}|}$$

where $J_{t_i} = Card\left(I_{t_i}\right)$ and $I_{t_i} = \{I_j \mid t_i \in I_j\}$
Hence, the tf-idf score is given by

$$tf - idf_{ij} = tf_{ij}.idf_i$$

Note that the idf score can be computed after the clustering and the construction of the inverted file, while it is fixed for any test image which is further described.

3.2 Classification

Two widely known classification schemes are the random forest and the support vector machines (SVM). The success of random forest in recent times has increased its popularity. Application of random forest classifier in the work of [36] has made great advances in human pose estimation. SVM is one of most widely used classifier in various problems [34] [13]. In this section the details of Random forest and SVM classifiers are explained.

3.2.1 Random Forest

Random forest has proven fast and effective multi-class classifier for many problems such as human pose estimation [36], key point recognition [22], image categorization [37], fast codebook building [28]. Many other applications are shown in [12]. The random forest classifier is easy and fast to test. Its natural ability of multi-class classification makes it more powerful. It build on three basic ideas: 1) first decision trees algorithm by Quinlan [31], 2) Bootstrap Aggregation (Bagging) by Breiman [5] and 3) random selection of a small subset from all attributes to be tried on each node by Amit *et al*. [2].

Decision Tree

Classification and regression trees were introduced by Breiman in [4]. The most popular algorithm of decision trees is "C4.5" by Quinlan [31], which is described below. Decision tree is special type of graph, which has decision nodes and leaf nodes (Class nodes) as shown in figure 3.1.

- Each decision node tests an attribute.
- Each branch corresponds to an attribute value.
- Each leaf node assigns a classification(Class).

The decision attribute on each node is chosen based on the minimization of Shannon entropy and maximization of the information gain based on information theory aspects [35]. Let us assume that we are given set of N training samples $S = \{s_1 ... s_i ... s_N\}$. Each sample $s_i = \{X_i, Y_i\}$ has K attributes (X_i), in our case, the histogram of words and a class label (Y_i), where $X_i = \{x_1 .... x_k ... x_K\}$ is the histogram. The optimum attribute x^* is selected according to

$$x^* = \underset{x \in \tau}{\operatorname{argmin}} \sum_{S(x) \in (S_L, S_R)} \frac{|S(x)|}{|S|} I(S(x))$$

where τ is the set of values of given attribute x.

$I(S(x))$ is the Shannon entropy of set $S(x)$ after its division into S_L, S_R by x. If the set S is not classified correctly, algorithm recurs on the samples of S_L and S_R for the left and right child nodes respectively. Training the the decision tree is most expensive process of training, because every attribute must be tried on every node and the number of nodes grows exponentially as the tree's depth increases. Each leaf node is assigned a *prediction* probability $p_l(c)$ of each class (c) based on training samples that end up at the particular leaf.

Whereas our classification objective is to minimize the Shannon entropy $I^{cls}(S)$ of S for all the classes (cls). Shannon entropy for C classes (cls) in sample set S is given by

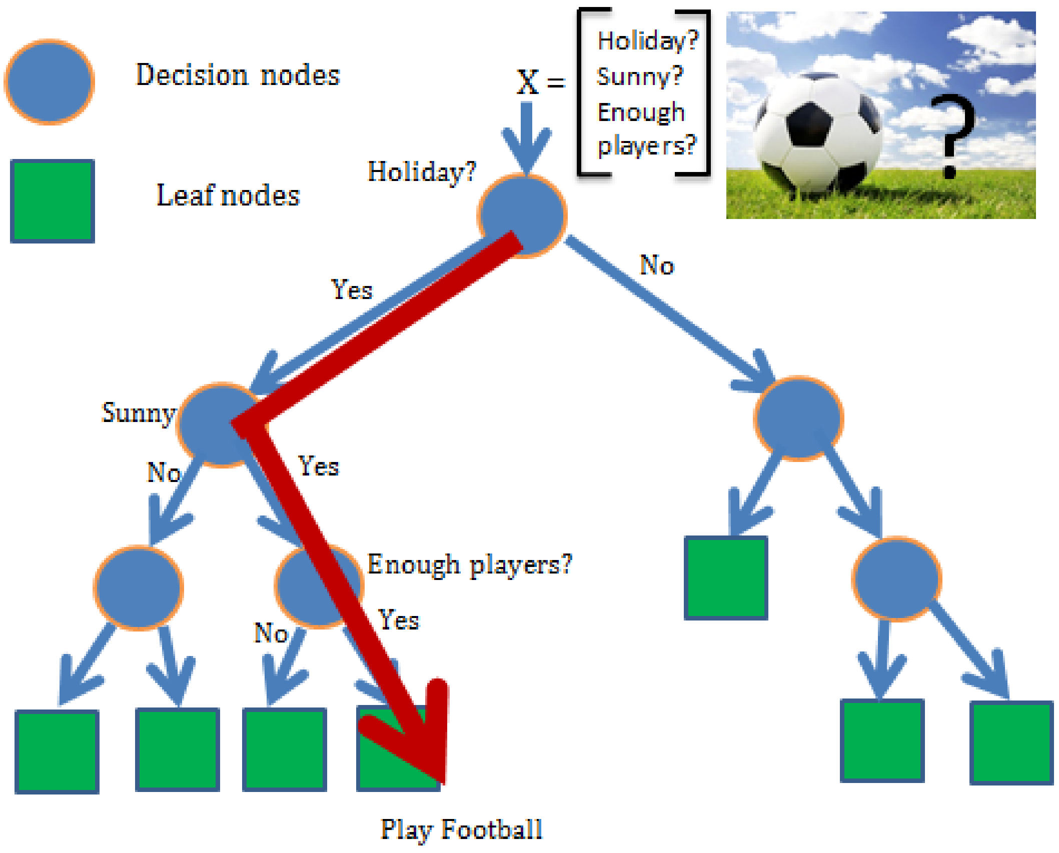

Figure 3.1: A simple example of Decision tree is shown, which shows the path followed for decision making. Where decision is made based on binary attributes (is it holiday?, is it sunny? and do we have enough players to play?). We can see from figure, if day is holiday and sunny and there are enough player to football then one would play football. Attributes on each node is chosen, based on its discriminative quality.

$$I^{cls}(S) = -\sum_{c=1}^{C} p(c \mid S) \log p(c \mid S)$$

Given a test sample, it is passed through the tree. It starts at the root node and ends up at a leaf node. The sample is assigned the *prediction* probability $p_l(c)$ of the leaf node. If tree is trained to full depth or perfect classification of training data samples, then it overfits the training data and has high variance on test data. There have been proposed methods to avoid over-fitting like pruning schemes [27].

Bagging

Given a training set S, a forest of multiple trees is built by using bootstrap sampling. Training samples for a tree are drawn by uniform sampling with replacement. About one-third of the cases that are left out of the samples are called out-of-bag (oob) samples. The oob sample is used to get an unbiased estimate of the classification error as trees are added to the forest. The oob samples are also used to determine the importance of variables. Trees are trained to full depth without any pruning. Bagging helps to avoid over-fitting by not using all training samples to train a tree. Instead, multiple trees are built by sub-sampling the training data. Consequently, sampling brings the stability to decision trees and decreases its variance on test data.

Random sampling of attributes

Though bagging brings stability to decision tree, training of many trees is still expensive, as every attribute has to be tried on each node to get the best split of samples. Amit *et al.* [2] proposed the idea of random sampling of attributes to be tried on every node, which makes this algorithm more fast. They propose to use the mean of all trees probabilities as the output of multiple trees.

Breiman in [6] consolidated all the work and injected another randomness by random sampling of training samples for the input of tree training. Final classification is made by combining the output probabilities p_l of each tree as

$$p(c \mid s) = \frac{1}{T}\sum_{t=1}^{T} p_l\,(c)$$

where T is total number of trees and s is test sample. Class with maximum probability $p(c \mid s)$ is chosen as output class.

3.2.2 Support Vector Machine

Support vector machine (SVM) [10] has been widely used for classification in several computer vision applications such as object recognition, human action recognition. A SVM for binary classification task will take as input a set of vectors and classify each of these vectors as belonging to one class or another. Given a set of labeled training examples, a binary SVM will learn a model that will maximize the separation between the two classes. Any new example is classified as one of the two classes.

Therefore, given N training examples $S = \{s_1...s_i...s_N\}$ SVM will find a hyperplane in a high-dimensional feature space which will maximize the separation or margin between the two classes. Each training example is defined as $s_i = \{X_i, Y_i\}$, where Y_i is the class label and X_i is the histogram of visual words. This hyperplane for a binary classification is estimated by optimizing the following problem defined in [10]:

$$\begin{gathered}\min_{w,\zeta,b}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\zeta_i\right\} \\ \text{s.t. } Y_i(wX_i - b) \geq 1 - \zeta_i, \zeta_i > 0\end{gathered} \tag{3.1}$$

where w is the weight vector to be estimated which parametrizes the hyperplane; ζ are the slack variables, for training data points which cannot be correctly classified i.e. are inside the margin and C is a parameter that controls the trade-off between the slack variables and the margin. The above problem can be solved by standard quadratic programming techniques.

Classification of human activities is inherently a multi-class problem i.e. the input vectors X_i has to be classified as one of the C classes $Y_i \in \{1,\ldots,c,\ldots,C\}$. Until now, we explained the SVM classification as a binary classification problem. In this paragraph we will summarize multi-class classification with SVMs. A way to perform multi-class classification with SVM is to reduce the multi-class classification into several binary classification problems. Two of the reduction methods are *one-versus-all strategy* and *1 vs 1 strategy*. In *one-versus-all strategy*, binary classifiers for each class, are build with one class as positive class and the rest as negative class. The label assigned to an unknown example is of the binary classifier which gives the highest output or *SVM score*. In *1 vs 1 strategy* an unknown example is assigned a class label which gets the maximum votes. Cramer *etal*. [11] cast the multi-class classification into a single optimization problem instead of dividing it into several binary classification problems.

Chapter 4

Feature representation for action recognition

We considered three types of feature representations , which are built from raw 3D points. These representations directly provides the descriptors of each depth image. we refer to these representation as :

1. Single-reference representation
2. Multi-reference representation
3. No-reference representation

In the next sections, we present in detail each of these representations.

4.1 Single reference representation

Inspired from [32], a single reference representation is used. Each $3D$ point in depth frame reflects a feature point if its depth is greater than zero. The descriptor is relative position to a reference point. This representation have many advantages, such as it characterizes the actor's pose, spatial position of body parts is preserved and it is depth invariant. When clustering is done then cluster centers corresponds to body parts as shown in Figure 4.1.

Actor's upper body position is taken as reference point in our case. Upper body is detected using the robust upper-body detection algorithm of [39], which uses Haar like features on grayscale images. If the upperbody detection algorithm fails in a frame then previous frame's detected position is used. Any other reference point can be used, such as head center or center of shoulders instead of the upper-body center position.

From the bounding box provided by the upper body detector, the center position of upper body part can be computed. This position is considered as reference point $f_r = (x_r, y_r, z_r)$. As mentioned 3D points in depth images are considered as feature points if their depth is greater than zero. If $f_i = \left(x_i, y_i, z_f\right)$ is the i^{th} feature point in a depth image, with z being the depth of the point (x_i, y_i), then the descriptor of this point is $d_i = (x_i - x_r, y_i - y_r, z_i - z_r)$.

If a depth image contains n valid features, then the whole set of descriptors can be represented by a $n \times 3$ matrix D as

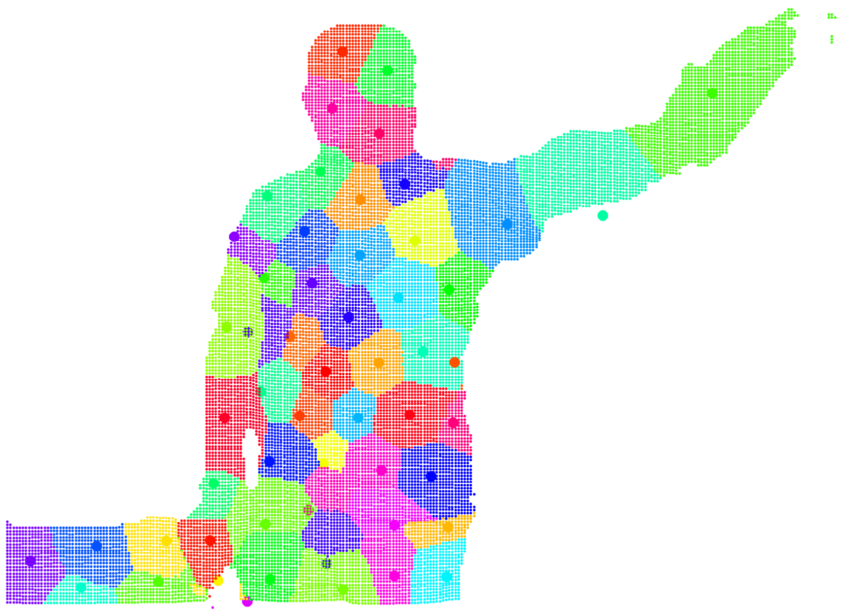

Figure 4.1: Representation of a frame from a test sequence using the single-reference representation. It can be seen that cluster centers corresponds to body parts

$$\mathbf{D} = \begin{bmatrix} d_1 \\ . \\ . \\ d_i \\ . \\ . \\ d_n \end{bmatrix} = \begin{bmatrix} x_1 - x_r, y_1 - y_r, z_1 - z_r \\ \dots \\ \dots \\ x_i - x_r, y_i - y_r, z_i - z_r \\ \dots \\ \dots \\ x_n - x_r, y_n - y_r, z_n - z_r \end{bmatrix}.$$

Given a visual vocabulary of length K each d_i is quantized and the matrix is mapped to a vector in the K-dimensional space.

4.2 Multi-reference representation

A representation of $3D$ points in depth frame, which does not require detection of any reference point is proposed. Pairwise relative position of human skeleton joints is used in [42], which promises more discriminative features. We sample the depth frame by uniformly sampling the data in both axis of depth image and sampled points are kept as feature points. The relative position of pairwise features is computed by taking the difference between the positions of the feature points. Given n feature points the position of the i^{th} feature point f_i with respect to the

j^{th} feature point f_j is computed and produces a new *relative feature* (descriptor) d_{ij}, which is computed as follows

$$d_{ij} = f_i - f_j,$$

where $f_i = (x_i, y_i, z_i)$, $f_j = \left(x_j, y_j, z_j\right)$, the coordinates of the two feature points. Hence, d_{ij} can be represented by

$$d_{ij} = \left(x_j - x_j, y_i - y_j, z_i - z_j\right).$$

If n feature points are sampled in a depth image then the final number of relative features is n^2. All the relative features can be represented in a $n \times n$ cell matrix, with each cell being a descriptor. But only upper triangular part of the matrix is kept, because lower triangular points are just the mirror points of upper triangular points and does not add to recognition performance. Thus, only relative features are kept and stacked into $\left(\frac{n^2-n}{2}\right) \times 3$ descriptors matrix $\mathbf{D}$ as shown below.

$$\mathbf{D} = \begin{bmatrix} x_1 - x_2, y_1 - y_2, z_1 - z_2 \\ \cdots \\ \cdots \\ x_1 - x_i, y_1 - y_i, z_1 - z_i \\ \cdots \\ \cdots \\ x_1 - x_n, y_1 - y_n, z_1 - z_n \\ x_2 - x_3, y_2 - y_3, z_2 - z_3 \\ \cdots \\ \cdots \\ x_i - x_j, y_i - y_j, z_i - z_j \\ \cdots \\ \cdots \\ x_{n-1} - x_n, y_{n-1} - y_n, z_{n-1} - z_n \end{bmatrix}$$

where j is always greater than i.

Sampling is the only parameter to be adjusted. It should be dense enough to pick points at key body parts, like arms, legs, etc, but not too dense to keep the computational complexity of algorithm to feasible limits on big datasets. Note that computational complexity of the algorithm grows exponentially with the size of the $\mathbf{D}$. We experimentally tried different sampling factors and their performance is discussed in chapter 6.

4.3 No-reference representation

The previous two representations are based on global translation of all 3D points, so it makes them depth invariant. To make descriptors locally invariant to similarity transformations a geometric hashing technique is used from [15] for video synchronization. This technique was originally presented by an application called *astrometry*, `http://astrometry.net/`, where quadruplets of nearby stars called *quads* are used for indexing night sky images.

Let us assume that $g(x,y)$ denotes the depth of the pixel (x,y). Using a radius δ, we consider its four neighborhood $(x-\delta, y)$, $(x+\delta, y)$, $(x, y-\delta)$ and $(x, y+\delta)$. We refer to this

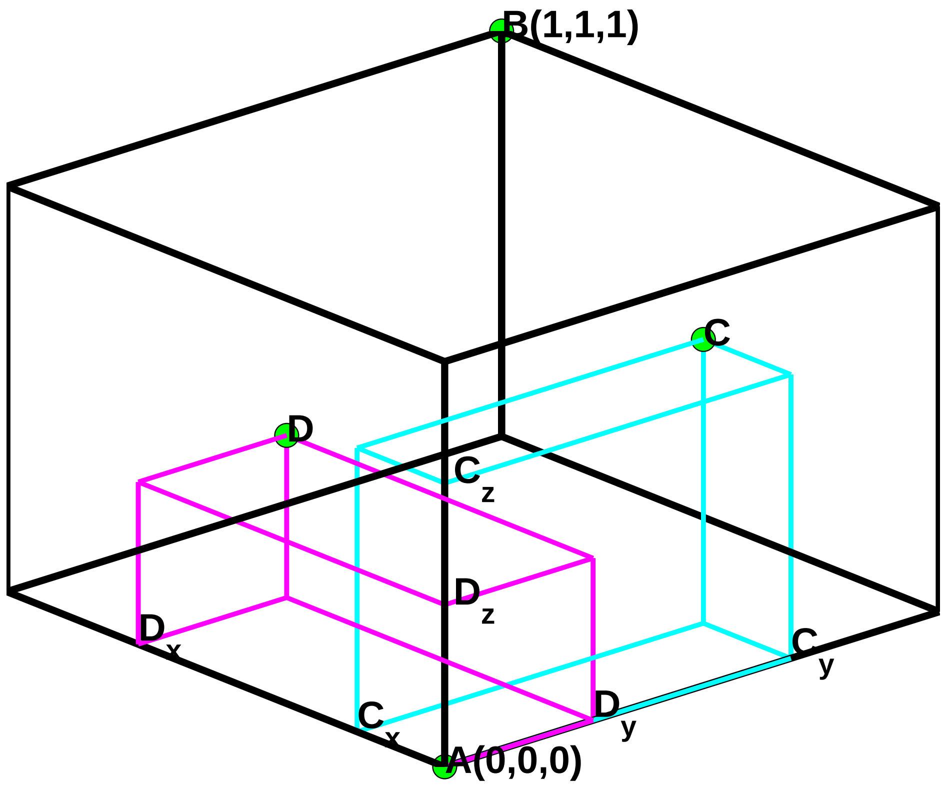

Figure 4.2: Coordinate system defined by two widely separated points (AB). 3D coordinates of the points C and D are encoded as descriptor in local coordinate system defined two most widely separated points A and B.

neighborhood as δ-neighborhood of point (x,y). Together with the depth values $g(x-\delta,y)$, $g(x+\delta,y)$, $g(x,y-\delta)$ and $g(x,y+\delta)$, we obtain a quadruple of 3D points.

If we denote A,B,C,D as the δ-neighborhood of a point (x,y) with (A,B) the pair of two most widely separated points in the quad, we we can consider a local 3D coordinate system such that A coincides with $(0,0,0)$ and B coincides with $(1,1,1)$. This way we can encode the relative position of the points C and D with respect to the points A, B as shown in Figure 4.2. Finally, the position of C and D in local coordinate system are kept as six dimensional dimensional descriptor that is $d = (C_x, C_y, C_z, D_x, D_y, D_z)$, which is shown in figure 4.2.

In other words, a similarity transformation applies to each quadruple defined through the δ-neighborhood of a feature point. The feature points can be all pixels of a depth frame (excluding the pixels near the image boundary) or a subset of them after a regular sampling. The permutation of the points in the quad causes a kind of reflection [15]. To cancel this, given the pair (AB), we consider the point with the smallest depth value as the origin of local coordinate system. Moreover, given the pair (CD) we first describe the coordinates the point more close to the origin. From now on, we refer to this as quad descriptor, or simple quad.

Chapter 5

BOW representation of frames and video

Inspired from the success of Bag of words approach in image classification, we describe each frame by a BOW representation from our frame descriptors. Once frame descriptors are computed, they are put into a BOW framework as described in chapter 3. We recall that d denotes a feature point descriptor and D denotes all frame descriptors stacked in a matrix. This chapter is divided into three parts.

1. Codebook building
2. Frame BOW representation
3. Video BOW representation

5.1 Codebook building

The codebook is built using the descriptors of some training frames. If we have N training frames and D_i is the frame descriptors of each training frame, then they are put into one long array as

$$DT = \begin{bmatrix} D_1 \\ .. \\ D_i \\ .. \\ D_N \end{bmatrix}$$

DT contains all the descriptors from the training frames, which is very big and computational expensive to be quantized. Therefore, we only use 10% of training descriptors to build the codebook by randomly sampling the rows of DT. K-means with the Euclidean distance is used to do the clustering of training descriptors. This scheme provides a codebook, parametrized by the number of cluster centers (K). K-means is run with multiple initializations of cluster centers and the one with minimum clustering error provides the cluster centers, which are the visual words of codebook.

5.2 Frame BOW representation

Frame descriptors are quantized by their projection on the codebook. This way each frame can be represented by a histogram with K bins, where each element is frequency of the particular word in the frame. It is weighted using the $Tf - Idf$ weighting scheme as described in chapter 3. This way each frame is described by a K-length vector X. The variation of the visual words over a temporal window accounts for the motion information which is integrated in each frame BOW representation. Here we describe two methods to capture motion from BOW representation over a temporal window.

- Temporal smoothing of X
- Temporal differentiation of X

5.2.1 Temporal smoothing of X

To describe the temporal pose information, temporal smoothing is used by using Gaussian filter of scale σ_s and size $4\sigma_s + 1$. The representation of a video with M frames can be written as a $K \times M$ array $\mathbf{X} = \{X_1, X_2, ...X_m, ...X_M\}$. The smoothed vector X^s of a frame can be computed by the following convolution

$$X^s_{mk} = \sum_{i=-2\sigma_s}^{2\sigma_s} w_i X_{(m+i)k}$$

where w_i's are weights of the Gaussian filter and k stands for each word. The weights w_i's of the Gaussian filter are computed as follow

$$w_i = \frac{1}{\sqrt{2\pi}\sigma_s} e^{-\frac{i^2}{2\sigma_s{}^2}}, \qquad i = -2\sigma_s....2\sigma_s$$

Finally, each vector X_m is replaced by its smoothed version.

5.2.2 Temporal differentiation of X

To capture the motion change, the temporal derivative of the BOW representation$X$ can be also used. Using the same BOW representation of video $\mathbf{X} = \{X_1, X_2, ...X_m, ...X_M\}$, each vector with the derivatives X' can be computed by

$$X'_{mk} = \sum_{i=-2\sigma_s}^{2\sigma_s} w_i X_{(m+i)k}$$

Where w_i's are weights of Gaussian derivative filter and k stand for each word. The weights of the Gaussian derivative filter are computed by differentiating the above Gaussian filter

$$w_i = \frac{i}{\sqrt{2\pi}\sigma_s{}^3} e^{-\frac{i^2}{2\sigma_s{}^2}}, \qquad i = -2\sigma_s....2\sigma_s$$

Finally, once X^s and X' are computed they are normalized by their L1 norm. Optionally, one can use either vector or concatenate them in a $2K \times 1$ vector. We test all the cases in experimental part of the thesis in chapter 6.

Figure 5.1 show all three type of BOW words representations of frames over time in a video with multiple action classes.

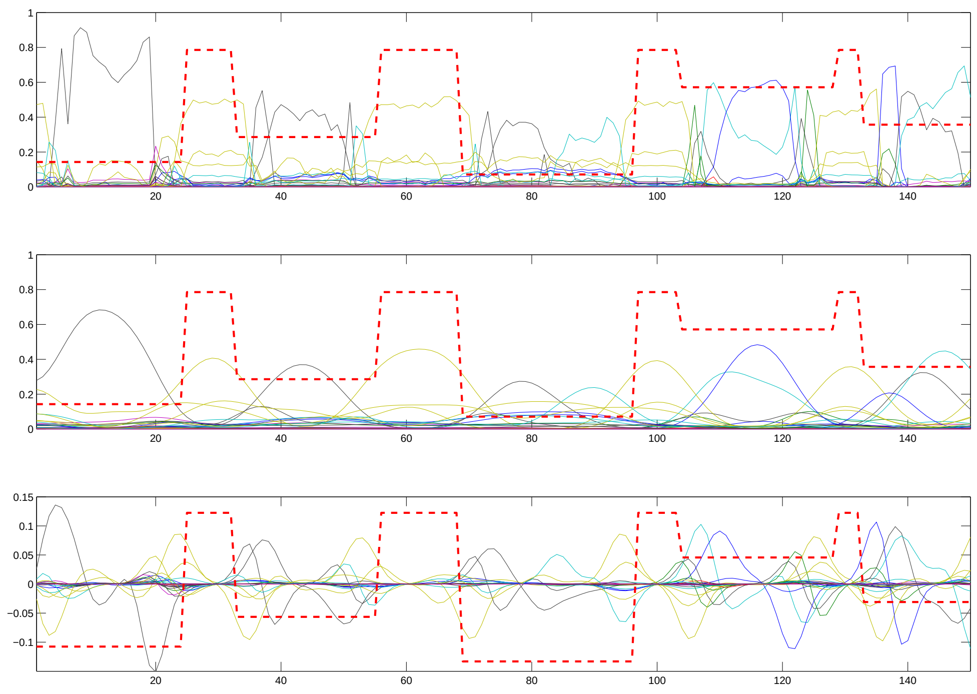

Figure 5.1: From top to bottom, frame BOW representation (X) of each frame of a video, Temporal smoothing of X (X^s) and Temporal differentiation of X (X') . In all the figure red doted line corresponds multiple action classes. We can see that different words provides the discriminative information about each class.

5.3 Video BOW representation

Similar to the frame representation a video can also be represented using a BOW scheme. If we have M frames in a video, then all frame descriptors are put into long array DV that is,

$$DV = \begin{bmatrix} D_1 \\ .. \\ D_m \\ .. \\ D_M \end{bmatrix}$$

Each row of the DV array is quantized by its projection on the codebook. So that DV is transformed to a compact representation of a video. We refer to it as video BOW representation X of video. Again we normalize X by its L1 norm.

Chapter 6

Experiments

Comparisons are very important in research, putting things side by side can give a perspective. We compare our method with state-of-the-art methods in the later part of the chapter. Also, we observe the difference in the performance of the random forest and linear SVM with the increase of training data.

This chapter is basically divided in two parts, 1) Analysis of our algorithm with different parameters of each feature representation approach and 2) State of the art comparison on a publicly available dataset. The outline of the chapter is given below.

1. Datasets
2. Results on CharLearn dataset
3. Results on MSR Action3D dataset
4. Comparison with state-of-the-art methods
5. Implementation details

6.1 Datasets

Our algorithm is evaluated on two datasets, one used for development purposes and the another one for state-of-the-art comparisons. These datasets are following:

- CharLearn dataset [1]
- MSR Action3D dataset [23]

6.1.1 CharLearn dataset

Part of *ChaLearn Gesture Dataset* [1] is used for development purposes. Because the whole dataset is very big (around 50k gestures), we only gathered a small subset (devel04), which includes 10 gestures performed by a single actor in front of a fixed depth camera (Kinect). This dataset contains both intensity and depth images aligned and synchronized, so upperbody parts can be easily detected. It includes 10 hand gestures, such as *point straight, stop sign, cross arm, point right, join hands in front, wave right hand, make fist on right, stop process action, take fist top to bottom*. It contains 47 videos and annotations are provided. A single video may

Action Set1 (AS1)	Action Set 2 (AS2)	Action Set1 (AS2)
Horizontal arm wave	Horizontal arm wave	High throw
Hammer	hand catch	forward kick
Forward punch	Draw x	side kick
High throw	Draw tick	jogging
Hand clap	Draw circle	Tennis swing
Bend	Two hand wave	Tennis serve
Tennis serve	Forward kick	Golf swing
pickup & throw	side boxing	pickup & throw

Table 6.1: The three subsets of actions used in experiments

contain one action or multiple actions (up to 7). Every video starts with an actor standing still and ends with the same pose, also in between the actions the actor stands still. In order to use all the videos, instead of just annotating some parts, we add another class *stand still*. Finally, we have 153 action sequences and 3127 frames, where each action is about 10-30 frames long. Each action is performed at least 8 times and at least two of them are kept for training data. We divide the data in training and test sets in the ratio of 1:4, which means that 31 actions are kept for training and the rest are used for testing.

6.1.2 MSR Action3D Dataset

MSR-Action3D dataset [23] is an action dataset of depth sequences. This dataset contains 20 actions: *horizontal arm wave, horizontal arm wave, hammer, hand catch, forward punch, high throw, draw x, draw tick, draw circle, hand clap, two hand wave, side-boxing, bend, forward kick, side kick, jogging, tennis swing, tennis serve, golf swing, pick up and throw*. Each action is performed 2-3 times by ten subjects. The subjects were facing the camera during the performance. The depth maps were captured at 15 frames per second by a depth camera (Kinect). The size of the depth map is 640×480. Altogether, the dataset has 23797 frames of depth maps for the 557 action samples. Each action video is from 35 to 85 frames long and 50 frames long on average.

Dataset is divided into three subsets, each having 8 actions. Table 6.1 lists the three action subsets used in the experiments. The subsets AS1 and AS2 contain actions with similar movement, and subset AS3 is composed of complex actions. We compare our algorithm with other state-of-the-art methods by following the same testing conditions as [23], [42], [44], where five subjects are used for training and other five for testing.

6.2 Results on CharLearn dataset

All the experiments on this dataset are done without removing the background. There is only one parameter to tune for random forest training, which is the number of trees. It is tuned using out-of-bag (oob) score as shown in Figure 6.1. We train 45 and 35 trees for frame-wise classification and isolated action classification respectively. We can see from Figure 6.1 that even if the number of trees increases after 30, the oob score and test score (accuracy on test set) remain almost the same. So, even if the number of trees are high, still it does not overfit.

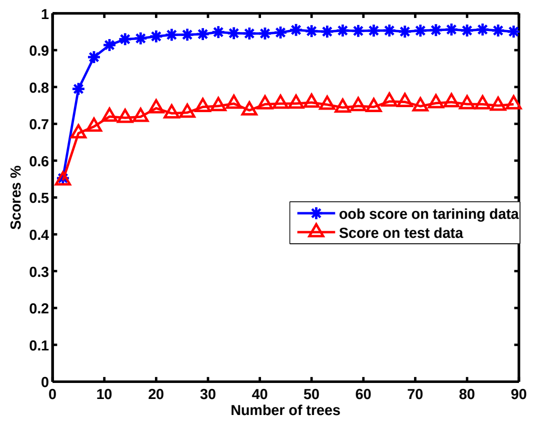

Figure 6.1: Variation of oob score and test score of frame-wise classification with the increase of number of trees using single-reference representation on CharLearn dataset. As the trees added to the forest oob score on training data become steady, we can see in figure that test score also become steady with oob score. So it means the oob score is unbiased classification estimate of the training set.

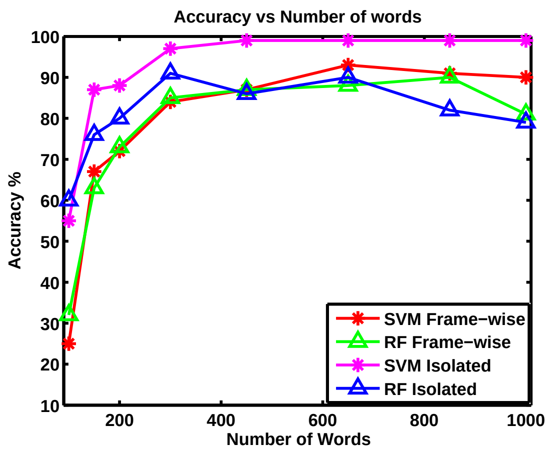

Figure 6.2: Performance of random forest (RF) and SVM for frame-wise and isolated classification with the variation of the number of words by using single-reference representation on CharLearn dataset

Also, Breiman in [6] claims that random forest does not overfit. Due to randomness involved in random forest, we run it 10 times and take the mean as final accuracy. Whereas, regularization constant (C) for linear SVM found by 2-fold cross validation on training data using LIBSVM [8]. Regularization constant is first searched in logarithmic scale and then uniformly around the best value of C in the initial cross validation. This process is repeated for all the tests to find a suitable regularization constant during cross validation.

Another parameter to tune is the window size for temporal derivatives and temporal smoothing of frame-wise BOW representations. However, since the window size is $4\sigma_s + 1$of frame where σ_s is the scale of the Gaussian kernel, we tune this parameter. It is observed that the scale (σ_s) equals to 2 for temporal derivatives and equals to 4 for temporal smoothing works

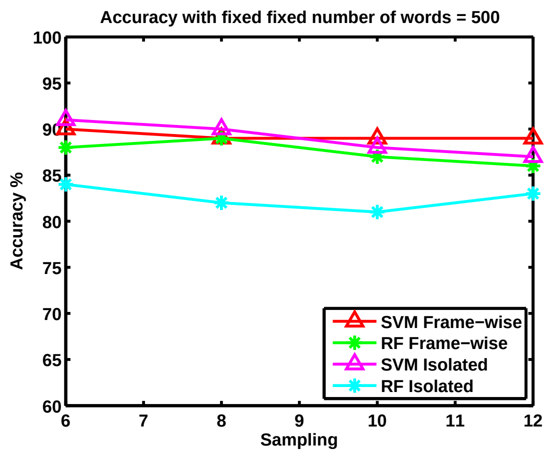

Figure 6.3: Performance of random forest (RF) and SVM for frame-wise and isolated classification with the variation of sampling. When number of words is fixed for multi-reference representation on CharLearn dataset.

well on this dataset. We use a vector obtained from the concatenation of temporal derivatives and temporal smoothing vectors of frame BOW representation of each frame. We make the comparison of frame representations: temporal derivatives of BOW, temporal smoothing of BOW and concatenation of both in next section. Now, we present the analysis of all the feature representation approaches.

1. Analysis of single-reference representation
2. Analysis of multi-reference representation
3. Analysis of no-reference representation

6.2.1 Analysis of single-reference representation

Visual words correspond to body parts as shown in chapter 4. As the number of words increase, the performance of frame-wise classification and isolated classification also increases as shown in Figure 6.2. It means that with the increase of number of words, the algorithm is able to capture the subtle movement of body parts. We can see from Figure 6.2 that frame-wise classification performance is lower than the isolated classification case, because frame-wise classification only takes into account the variation of BOW within a small temporal window, whereas isolated representation of action takes into account all the poses involved in a particular action. We also observe that most of the frame misclassification occurs on the boundaries of the actions.

6.2.2 Analysis of multi-reference representation

Sampling is a very crucial parameter to tune in this approach, because the computational complexity increases exponentially with the number of feature points sampled. We experiment with different sampling factors by varying it from 6 to 12 (factor means that every 6^{th} pixel of depth map is kept as feature point). So, sampling equal to 6 is more computationally expensive than

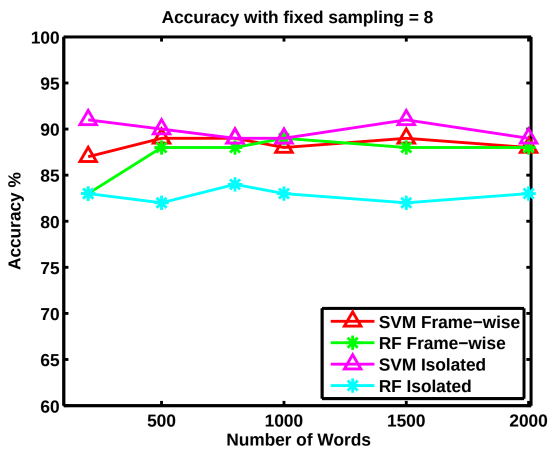

Figure 6.4: Performance of random forest (RF) and SVM for frame-wise and isolated classification with the variation of number of words. When sampling is fixed for multi-reference representation on CharLearn dataset.

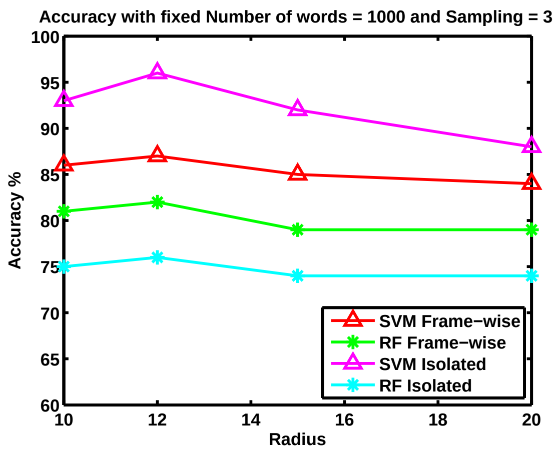

Figure 6.5: Performance of random forest (RF) and SVM for frame-wise and isolated classification with the variation of radius (δ), when sampling and number of words are fixed using no-reference representation on CharLearn dataset.

the one to 12. We can see from Figure 6.3 that when the sampling increases the performance decreases, but we also observe that this decrease in performance can be compensated for by the increase of the number of words, which can be seen from Figures 6.3 and 6.4 for sampling factor equal to 8.

6.2.3 Analysis of No-reference representation

The only parameter to tune in this approach is the δ-neighborhood. We experiment with different values of radius (δ), which is shown in Figure 6.5. We observe that radius equal to 12 works best for this dataset. We also experimented with uniform sampling of feature points. As the sampling factor increases the performance remains almost the same as shown in Figure 6.6. We also observe that if the feature points are chosen in a more sophisticated way then we

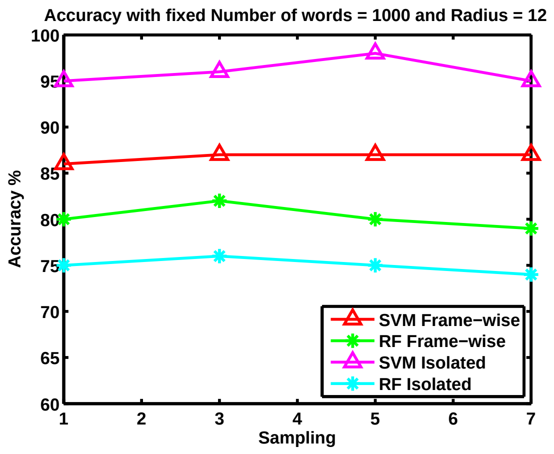

Figure 6.6: Performance of random forest (RF) and SVM for frame-wise and isolated classification with the variation of sampling, when radius (δ) and number of words are fixed using no-reference representation on CharLearn dataset

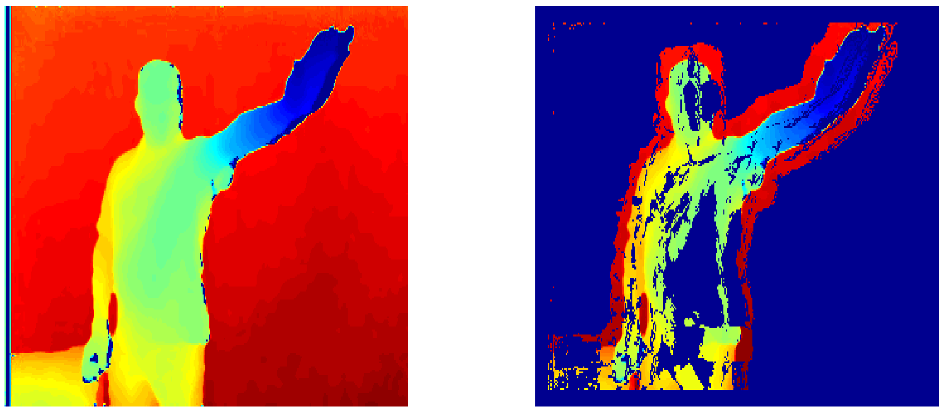

Figure 6.7: A depth map from test sequence on the left and in the right image, points with non zero idf are shown. which shows that points corresponding to the words that appears in all training images are removed.

do not need to compute quad descriptors for all pixels of the depth image and the performance may further increase.

We also show that background descriptors are very similar and appear in every frame of the training dataset, so the inverse document frequency for background words is zero. It is shown in Figure 6.7, that background descriptors of a test frame are removed, because they are projected on to the words that appear in all frames of the training data.

It can be seen from the Figures 6.2, 6.4 and 6.5 that single-reference performed best and then multi-reference approach, same is shown in Table 6.2.

Methods	SVM Frame-wise	SVM Isolated	RF frame-wise	RF isolated
Single-reference	92.50%	99.12%	89.50%	92.50%
Multi-reference	89.45%	92.30%	90.05%	84.67%
No-reference	87.10%	96.60%	82.09%	75.30%

Table 6.2: Comparison of frame representation approaches on CharLearn dataset.

6.3 Results on MSR Action3D dataset

MSR Action3D dataset contains only depth maps with the background being already removed, while we have already shown that our algorithm performs well with background in previous dataset. This dataset is challenging because it contains very similar actions , especially in AS1 and AS2 subsets. We only test multi-reference representation and no-reference representation on this dataset, because intensity images are not provided for upperbody detection and, moreover, skeleton joints provided with the dataset are not aligned with depth maps. So we can not use single-reference representation.

Numbers of trees are tuned similarly to previous dataset as shown in Figure 6.1 by using oob scores as trees are added to the forest. Regularization constant (C) for the SVM is found by 5-fold cross validation on training data as there are five actors in training set. Each video has only one action, so, for the isolated action recognition each video is described by a video BOW representation. We observed that scale (σ_s) equal to 9 for derivatives and σ_s equal to 12 for temporal smoothing works well on the MSR Action3D dataset. Frame-wise approach is also used to classify each frame. Once each frame is classified, then one can do the voting using class labels of each frame and chose the class that has maximum number of frames labeled in a video. We will refer to it as *voting* scheme. We observe that voting improves the isolated classification performance, which can be seen in Figures 6.8 and 6.9. Further, this section is divided into two parts as follows

1. Random forest vs SVM
2. Comparison of different frame BOW representations

6.3.1 Random forest vs SVM

We use concatenated BOW representations of frame to compare the performance of the SVM and random forest classifier. Same frame BOW representation is used on CharLearn dataset. We observe from Figure 6.3 and 6.4 that SVM outperforms random forest in both isolated and frame-wise classification case on CharLearn dataset using multi-reference representation. This drop in the performance of the random forest is observed in all three frame representations on CharLearn dataset. On the other hand, random forest outperforms SVM on frame-wise classification on MSR Action3D dataset, which is shown in Figure 6.8. Whereas, in the case of isolated action recognition case both random forest and SVM perform very similarly. It is shown in [7] that random forest works better on low dimensional feature than SVM and SVM works better when the dimensionality of feature space is large. In our case, the dimensionality of feature space is small but we have very few training examples in CharLearn dataset to train random trees properly. Thus, these interesting results lead us to believe that linear SVM

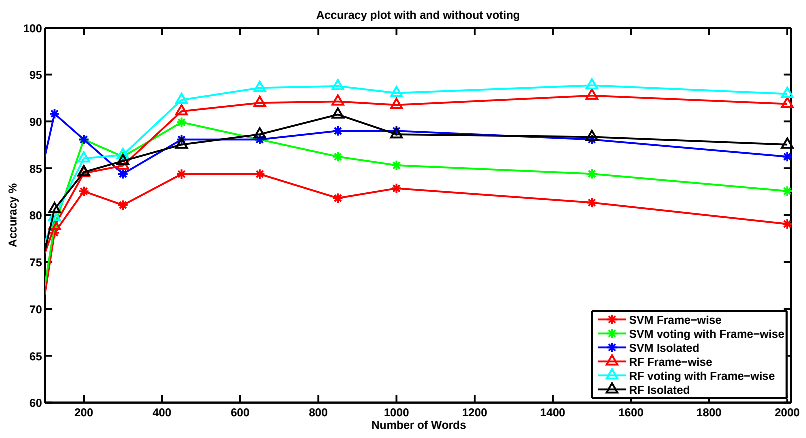

Figure 6.8: Frame-wise and isolated recognition accuracy with the variation of the number of words. Both Random forest (RF) and SVM are used. Multi-reference representation used for sampling factor equal to 10 on MSR action 3D dataset . Random forest out perform SVM in frame-wise classification. SVM and random forest perform similar in the case of isolated classification

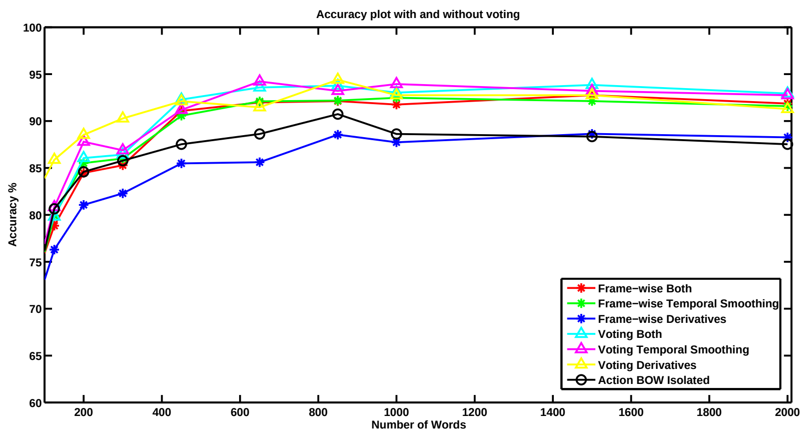

Figure 6.9: Frame-wise and isolated recognition accuracy with voting and BOW representation on MSR Action3D dataset, when number of words vary and sampling is fixed (fixed =10). Random forest is used on multi-reference representation. Temporal smoothing and temporal derivatives of BOW perform very similar to concatenation of both.

performs better than random forest when training data is small, as our feature space is low dimensional so random forest works better than SVM when it has enough training examples in MSR Action3D dataset.

6.3.2 Comparison of BOW representations

Once temporal smoothing of BOW (X^s) and temporal derivatives of BOW (X') representation is obtained, then one can use either vector or concatenate them. It is shown in Figure 6.9, that X^s and X' alone performs very similar, but when both concatenated, there is an increment in

Test set	SVM Frame-wise	SVM Isolated	SVM with voting	RF Frame-wise	RF Isolated	RF with voting	Li *etal* [23]	Xia *etal* [45]
AS1	86.26%	90.83%	89.91%	92.75%	90.73%	93.83%	72.90%	87.98%
AS2	78.81%	82.20%	83.10%	82.41%	82.20%	86.29%	71.90%	85.48%
AS3	89.17%	93.83%	95.58%	89.40%	90.09%	92.83%	79.20%	63.46%
Overall	84.75%	88.94%	89.53%	88.16%	87.67%	90.98%	74.70%	78.97%

Table 6.3: Recognition results of multi-reference representation method. In this table we compare with Li *etal* [23] and Xia *etal* [45]. Our algorithm outperforms the two methods in all cases

Test set	SVM Frame-wise	SVM Isolated	SVM with voting	RF Frame-wise	RF Isolated	RF with voting	Li *etal* [23]	Xia *etal* [45]
AS1	71.25%	81.65%	78.90%	83.08%	73.00%	85.14%	72.90%	87.98%
AS2	68.97%	71.81%	73.64%	68.84%	63.55%	73.55%	71.90%	85.48%
AS3	76.37%	81.42%	81.42%	87.97%	86.81%	89.65%	79.20%	63.46%
Overall	72.19%	78.29%	77.99%	79.96%	79.96%	82.78%	74.7%	78.97%

Table 6.4: Recognition results of no-reference representation method and comparison with Li *etal* [23] and Xia *etal* [45].

performance.

6.4 State-of-the-art comparison

We compare our method with the state-of-the-art methods of Wang *etal* [42], Li *etal* [23], Oreife *etal* [29], Xia *etal* [44] and Xia *etal* [45]. We use concatenated vector of X^s and X' for our algorithm as it performs the best.

We can see that our algorithm achieves the highest isolated action recognition accuracy using voting in Tables 6.3, 6.4 and 6.5. Multi-reference representation performs better than the state-of-the-art method, which can be seen in Tables 6.3 and 6.5. Whereas no-reference approach also has comparable results, which can be seen in Tables 6.4 and 6.5.

6.5 Implementation

I implemented an action recognition system according to the methods described in previous chapters. The system and the experiments were done by combining various programming languages, Matlab, Python and R [18]. R is used for fast K-means computation, and Python for random forest classification and upper-body detection. Rest of the work was done in Matlab. I used the OpenCV [3] function for upperbody detection. Scikit-learn [30] provided me help with the implementation of random forest. I used LIBSVM [8] for the cross validation as well as for classification task for linear SVM. However, if the system has to be embedded onto a

Methods	Overall accuracy %
Li *etal* [23]	74.70
Xia *etal* [45]	78.97
Wang *etal* [42]	88.20
Oreife *etal* [29]	88.89
Xia *etal* [44]	89.30
Multi-reference, RF with voting	90.98
Multi-reference, RF Isolated	87.67
Multi-reference, SVM with voting	89.53
Multi-reference, SVM Isolated	84.75
No-reference, RF with voting	82.78
No-reference, RF Isolated	79.96
No-reference, SVM with voting	77.99
No-reference, SVM Isolated	78.29

Table 6.5: Comparison of state-of-the-art methods with our multi-reference representation and no-reference representation methods. Multi-reference representation outperforms all the other methods, when voting is done after the frame-wise classification.

robot, then it would have to be implemented in a low-level programming language such as C or C++.

Chapter 7

Conclusion

We present frame-wise vector representations using BOW scheme build from three types of feature representations. We show that variation of the visual words over a temporal window accounts for the motion information which is integrated in each frame BOW representation. The feature representations are built from raw 3D points. These representations directly provide the descriptors of each depth image. The novelty of these representations is that they do not require to detect skeleton model or segmentation of the depth image, which is the bottleneck of the most approaches discussed in the state-of-the-art chapter 2. Moreover, frame representations intrinsically encodes the human pose information, which highlights the use of depth camera. We are able to classify almost 85% frames correctly using our multi-reference representation and almost 80% using no-reference representation on MSR Action3D dataset. Single-reference representation can be very useful; if one can detect a single reference point on human body such as face, head, center of shoulder etc.

Frame-wise classification is rich in possibility for extensions. As shown in the experiments, simple voting scheme after frame-wise classification for isolated action recognition gives better results than video classification based on video BOW representation. But it is possible to use any global optimization scheme for simultaneous video segmentation and action recognition after frame-wise classification. We apply a simple voting scheme after frame-wise classification for isolated action recognition to compare our method with state-of-the-art methods on isolated action recognition and we able to achieve better performance than state-of-the-art methods. We also compared the random forest and SVM on both datasets used in the experimentation. It is shown in the experiments that random forest performs better, when number of training examples are high and dimensionality of feature space is low. On the other hand, SVM performs well even if the number of training samples are low.

Bibliography

[1] Chalearn gesture dataset (cgd2011), chalearn, california, copyright (c) chalearn, 2011. `http://gesture.chalearn.org/`.

[2] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588, 1997.

[3] Gary Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[4] Leo Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[5] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[9] Lulu Chen, Hong Wei, and James M Ferryman. A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, 2013.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[12] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2-3):81–227, 2011.

[13] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, page 22, 2004.

[14] Piotr Dollar, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005*, pages 65–72.

[15] G Evangelidis and Christian Bauckhage. Efficient subframe video alignment using short descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2013.

[16] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[17] Minh Hoai, Zhen-Zhong Lan, and Fernando De la Torre. Joint segmentation and classification of human actions in video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3265–3272, 2011.

[18] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[19] J. Cech K. Kulkarni, G. Evangelidis and R. Horaud. Continous action recognition based on dynamic programming. *Submitted to International Journal of Computer Vision.*

[20] Alexander Klaser and Marcin Marszalek. A spatio-temporal descriptor based on 3d-gradients. In *IN 19th British Machine Vision Conference, Sep 2008*, volume 275, pages 1–10, 2008.

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

[22] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, volume 2, pages 775–781.

[23] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2010*, pages 9–14.

[24] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Expandable data-driven graphical modeling of human actions based on salient postures. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1499–1510, 2008.

[25] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[26] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

[27] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

[28] Frank Moosmann, Bill Triggs, Frederic Jurie, et al. Fast discriminative visual codebooks using randomized clustering forests. *Advances in Neural Information Processing Systems 19*, pages 985–992, 2007.

[29] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the 24th IEEE Conference on, Computer Vision and Pattern Recognition (CVPR)*, 2013.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] John Ross Quinlan. *C4.5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[32] Jordi Sanchez-Riera, Jan Cech, and Radu P. Horaud. Action recognition robust to background clutter by using stereo vision. In *The Fourth International Workshop on Video Event Categorization, Tagging and Retrieval*, LNCS. Springer, October 2012.

[33] Konrad Schindler and Luc Van Gool. Action snippets: How many frames does human action recognition require? In *Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8.

[34] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, ICPR, 2004*, volume 3, pages 32–36. IEEE.

[35] Claude Elwood Shannon. *A mathematical theory of communication*. American Telephone and Telegraph Company, 1948.

[36] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. *CVPR*, 2:3, 2011.

[37] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.*, pages 1–8.

[38] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *In Proceedings of Ninth IEEE International Conference on Computer Vision, 2003.*, pages 1470–1477, 2003.

[39] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[40] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, Cordelia Schmid, et al. Evaluation of local spatio-temporal features for action recognition. 2009.

[41] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3d action recognition with random occupancy patterns. In *Computer Vision–ECCV 2012*, pages 872–885. Springer, 2012.

[42] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1290–1297. IEEE, 2012.

[43] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[44] Lu Xia and JK Aggarwal. Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. *In Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, Oregon*, 2013.

[45] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 20–27, 2012.

[46] Xiaodong Yang and YingLi Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2012*, pages 14–19, 2012.

[47] Hao Zhang and Lynne E Parker. 4-dimensional local spatio-temporal features for human activity recognition. In *Intelligent Robots and Systems (IROS),*, pages 2044–2049. IEEE, 2011.