Journal of Machine Learning Research 14 (2013) 2549-2582 Submitted 1/13; Revised 5/13; Published 9/13

One-shot Learning Gesture Recognition from RGB-D Data Using Bag of Features

Jun Wan 09112088@BJTU.EDU.CN
Qiuqi Ruan QQRUAN@CENTER.NJTU.EDU.CN
Wei Li 08112050@BJTU.EDU.CN
Institute of Information Science
Beijing Jiaotong University
Beijing, 100044, China

Shuang Deng DAISY_SHUANG@HOTMAIL.COM
China machinery TDI international engineering co., Itd
Beijing, 100083, China

Editors: Isabelle Guyon and Vassilis Athitsos

Abstract

For one-shot learning gesture recognition, two important challenges are: how to extract distinctive features and how to learn a discriminative model from only one training sample per gesture class. For feature extraction, a new spatio-temporal feature representation called 3D enhanced motion scale-invariant feature transform (3D EMoSIFT) is proposed, which fuses RGB-D data. Compared with other features, the new feature set is invariant to scale and rotation, and has more compact and richer visual representations. For learning a discriminative model, all features extracted from training samples are clustered with the k-means algorithm to learn a visual codebook. Then, unlike the traditional bag of feature (BoF) models using vector quantization (VQ) to map each feature into a certain visual codeword, a sparse coding method named simulation orthogonal matching pursuit (SOMP) is applied and thus each feature can be represented by some linear combination of a small number of codewords. Compared with VQ, SOMP leads to a much lower reconstruction error and achieves better performance. The proposed approach has been evaluated on ChaLearn gesture database and the result has been ranked amongst the top best performing techniques on ChaLearn gesture challenge (round 2).

Keywords: gesture recognition, bag of features (BoF) model, one-shot learning, 3D enhanced motion scale invariant feature transform (3D EMoSIFT), Simulation Orthogonal Matching Pursuit (SOMP)

1. Introduction

Human gestures frequently provide a natural and intuitive communication modality in daily life, and the techniques of gesture recognition can be widely applied in many areas, such as human computer interaction (HCI) (Pavlovic et al., 1997; Zhu et al., 2002), robot control (Malima et al., 2006; Shan et al., 2007), sign language recognition (Gao et al., 2004; T. Starner and Pentland, 1998) and augmented reality (Reifinger et al., 2007). To model gesture signals and achieve acceptable recognition performance, the most common approaches are to use Hidden Markov Models (HMMs) or its variants (Kim et al., 2007) which are a powerful model that includes hidden state structure. Yamato

et al. (1992) used image preprocessing operations (background subtraction, image blurring) to extract low-level features and used HMM to recognize tennis motions. Brand et al. (1997) suggested a coupled HMM that combined two HMMs with causal possibly asymmetric links to recognize gestures. Vogler (2003) presented a parallel HMM algorithm to model gesture components and can recognize continuous gestures in sentences. Then a more general probabilistic model named dynamic Bayesian network (DBN) is proposed. DBN includes HMMs and Kalman filters as special cases (Suk et al., 2010). Youtian et al. (2006) defined five classes of gestures for HCI and developed a DBN-based model which used local features (contour, moment, height) and global features (velocity, orientation, distance) as observations. Suk et al. (2010) proposed a DBN-based system to control media player or slide presentation. They used local features (location, velocity) by skin extraction and motion tracking to design the DBN inference.

However, both HMM and DBN models assume that observations given the motion class labels are conditional independent. This restriction makes it difficult or impossible to accommodate long-range dependencies among observations or multiple overlapping features of the observations (Sminchisescu et al., 2005). Therefore, Sminchisescu et al. (2005) proposed conditional random fields (CRF) which can avoid the independence assumption between observations and allow non-local dependencies between state and observations. Wang et al. (2006) then incorporated hidden state variables into the CRF model, namely, hidden conditional random field (HCRF). They used HCRF to recognize gesture recognition and proved that HCRF can get better performance. Later, the latent-dynamic conditional field (LDCRF) model (Morency et al., 2007) was proposed, which combines the strengths of CRFs and HCRFs by capturing both extrinsic dynamics and intrinsic sub-structure. The detailed comparisons are evaluated by Morency et al. (2007).

Another important approach is dynamic time warping (DTW) widely used in gesture recognition. Early DTW-based methods were applied to isolated gesture recognition (Corradini, 2001; Lichtenauer et al., 2008). Then Ruiduo et al. (2007) proposed an enhanced Level-Building DTW method. This method can handle the movement epenthesis problem and simultaneously segment and match signs to continuous sign language sentences. Besides these methods, other approaches are also widely used for gesture recognition, such as linguistic sub-units (Cooper et al., 2012) and topology-preserving self-organizing networks (Flórez et al., 2002). Although the mentioned methods have delivered promising results, most of them assume that the local features (shape, velocity, orientation, position or trajectory) are detected well. However, the prior successes of hand detection and tracking are major challenging problems in complex surroundings. Moreover, as shown in Table 1, most of the mentioned methods need dozens or hundreds of training samples to achieve high recognition rates. For example, in Yamato et al. (1992), the authors used at least 50 samples for each class to train HMM and got the average recognition rate 96%. Besides, Yamato et al. (1992) suggested that the recognition rate will be unstable if the number of samples is small. When there is only one training sample per class, those methods are difficult to satisfy the requirement of high performance application systems.

In recent years, BoF-based methods derived from object categories (Fei-Fei and Perona, 2005) and action recognition (Wang et al., 2009) have become an important branch for gesture recognition. Dardas and Georganas (2011) proposed a method for real-time hand gesture recognition based on standard BoF model, but they first needed to detect and track hands and that would be difficult in a clutter background. For example, when the hand and face are overlapped or the background is similar to skin color, hand detection may fail. Shen et al. (2012) extracted maximum stable extremal regions (MSER) features (Forssen and Lowe, 2007) from the motion divergence fields which

paper/method	Kim et al., 2007/HMM	Yamato et al., 1992/HMM	Youtian et al., 1992/DBN	Suk et al., 2010/DBN	Sminchisescu et al., 2005/CRF
training samples per class	150	$\geq$50	15	42	NA
paper/method	Wang et al. 2006/HCRF	Morency et al. 2007/LDCRF	Corradini 2001/DTW	Lichtenauer et al. 2008/DTW	Ruiduo et al. 2007/DTW
training samples per class	$\geq$45	$\geq$269	45	$\geq$60	NA

Table 1: This tables shows the training samples pre class needed in some traditional methods. "NA" means the training samples are not clearly mentioned.

were calculated by optical flow (Lowe, 2004), and learned a codebook using hierarchical k-means algorithm, then matched the test gesture sequence with the database using a term frequency-inverse document frequency (tf-idf) weighting scheme. These methods need dozens or hundreds of training samples. However, in this paper, we explore one-shot learning gesture recognition (Malgireddy et al., 2012), that is, using one training sample per each class. Some important challenging issues for one-shot learning gesture recognition are the following:

1. **How to extract distinctive features?** Different people have different speeds, trajectories and spatial positions to perform the same gesture. Even when a single person performs the gestures, the trajectories are not identical. Therefore, the extracted spatio-temporal features should be invariant to image-plane rotation, scale and spatial position. Simple descriptors, such as motion trajectories (Yang et al., 2002) and spatio-temporal gradients (Freeman and Roth, 1995), may not meet the invariant conditions. Therefore, we propose a new spatio-temporal feature which is scale, image-plane rotation and space invariant and can capture more compact and richer visual representations. The new feature will be introduced in Section 3.1.

2. **How to select a suitable model?** Here, we select BoF-based model to recognize gestures because it reveals promising results for one-shot learning (Hernández-Vela et al., 2012) and has a number of attractive properties. First, in our BoF representation, we do not need the prior success of hand detection and tracking. Second, BoF is a modular system with three parts, namely, i) spatio-temporal feature extraction, ii) codebook learning and descriptor coding, iii) classifier, each of which can be easily replaced with different methods. For instance, we can apply various methods, such as Cuboid (Dollár et al., 2005) or Harris3D (Laptev, 2005) for the local spatio-temporal feature extraction while leaving the rest of the system unchanged.

In this paper, we focus on solving these two challenging issues and propose a new approach to achieve good performance for one-shot learning gesture recognition. Our experimental results reveal that our method is competitive to the state-of-the-art methods. The key contributions of the proposed method are summarized as follows:

- A new framework derived from the BoF model is proposed.
- A new spatio-temporal feature (3D EMoSIFT) is proposed.
- The new feature is invariant to scale and rotation.
- The new feature is not sensitive to slight motion.
- Using SOMP instead of VQ in the coding stage.

- Obtained high ranking results on ChaLearn gesture challenge.

The rest of paper is organized as follows: Section 2 reviews the background including BoF model and some local spatio-temporal features. In Section 3, we describe the proposed approach in detail. Section 4 presents the experimental results. In Section 5, we conclude the paper and discuss future work.

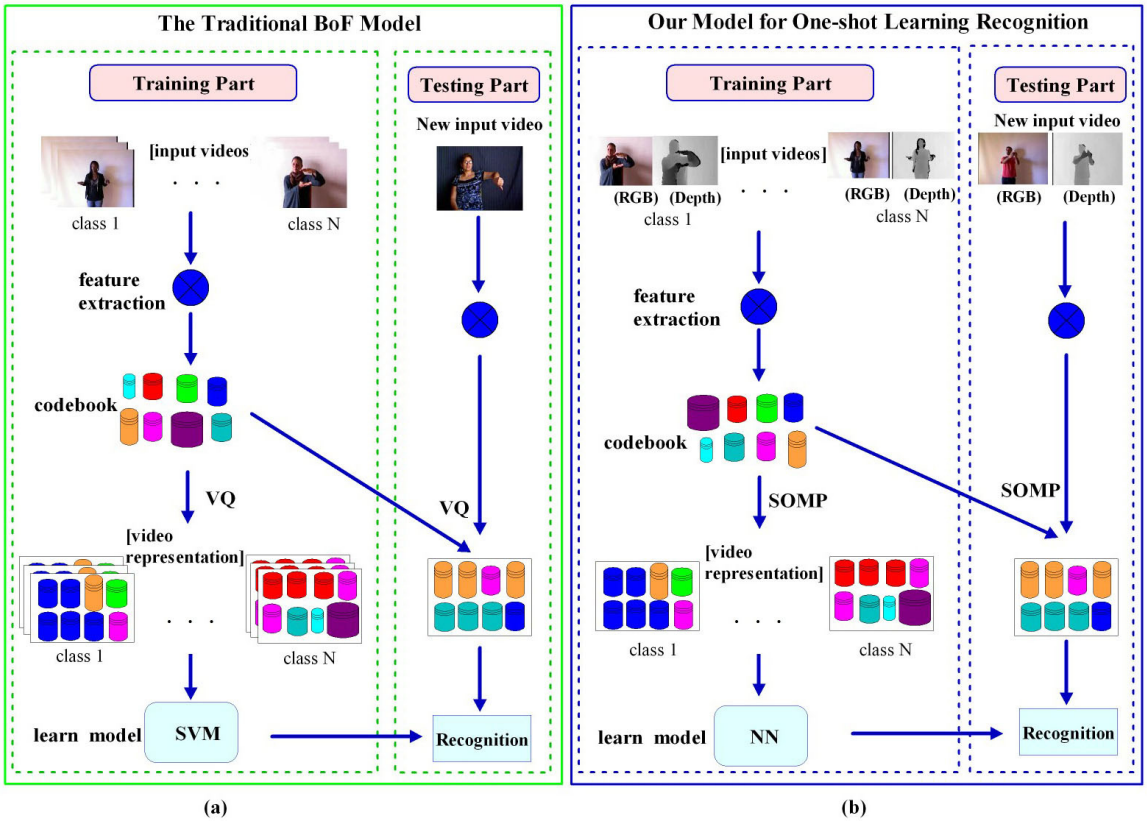

Figure 1: (a) An overview of the traditional BoF model (the left green rectangle); (b) An overview of our model (the right blue rectangle).

2. Background

In this section, we first introduce the traditional BoF framework for recognition and then review the local spatio-temporal features which are widely used in BoF model.

2.1 Traditional Bag of Feature (BoF) Model

Figure 1(a) illustrates the traditional BoF approach for gesture (or action) recognition. In the training part, after extracting local features from training videos, the visual codebook is learned with the k-means algorithm. Then each feature is mapped to a certain visual codeword through the clustering process and the video can be represented by the histogram of visual codewords. The histograms

representing training videos are treated as input vectors for a support vector machine (SVM) (Chang and Lin, 2011) to build a classifier. In the testing stage, the features are extracted from a new input video, and then those features are mapped into a histogram vector by the descriptor coding method (e.g., VQ) using the pre-trained codebook. Then, the histogram vector is finally fed into an SVM classifier to get the recognition result.

However, as shown in Figure 1(b), we list at least three differences between our model and the traditional BoF model. First, there is only one training sample per gesture class, while dozens or hundreds of training samples per class are provided in the traditional BoF model. Second, we use SOMP to replace VQ in the coding stage. That is because SOMP can get better performance. Third, in the recognition stage, we just use the simple nearest neighbor (NN) classifier instead of SVM to recognize gestures.

2.2 Spatio-Temporal Features

We describe some spatio-temporal features which represent the state-of-the-art techniques on object recognition tasks. Those features are commonly used to detect salient and stable local batches from videos.

The Cuboid detector depends on a set of linear filters for computing the response function of a video clip. The response function has the form of a 2D Gaussian smoothing function (applied in the spatial domain) and a quadrature pair of 1D Gabor filters (applied in the temporal direction). Then the keypoints are detected at the local maxima of the response function. The video batches extracted at each of the keypoints are converted to a descriptor. There are a number of ways to compute descriptors from video batches as discussed by Dollár et al. (2005). Among those, gradient-based descriptors such as histograms of oriented gradients (HOG) and concatenated gradient vectors are the most reliable ones. For more details about the Cuboid feature, please see Dollár et al. (2005).

The Harris3D detector (Laptev, 2005) is an extension of the Harris corner detector (Harris and Stephens, 1988). The author computes a spatio-temporal second-moment matrix at each video point using independent spatial and temporal scale values, a separable Gaussian smoothing function, and space-time gradients. The final locations of space-time interest points are given by the local positive spatio-temporal maxima. Then, at each keypoint, two types of descriptors are calculated, which are HOG and histograms of optical flow (HOF) descriptors.

The MoSIFT (Chen and Hauptmann, 2009) is derived from scale invariant feature transform (SIFT) (Lowe, 2004) and optical flow (Lucas et al., 1981). First, a pair of Gaussian pyramids are built from two successive frames, respectively. Then, optical flow pyramids are calculated by each layer of the pair of Gaussian pyramids. Next, a local extreme detected from difference of Gaussian pyramids (DoG) can only become an interest point if it has sufficient motion in the optical flow pyramid. Finally, as the process of the SIFT descriptor calculation, the MoSIFT descriptors are respectively computed from Gaussian pyramid and optical flow pyramid so that each MoSIFT descriptor now has 256 dimensions.

Ming et al. (2012) propose a new feature called 3D MoSIFT that is derived from MoSIFT. Compared with MoSIFT, 3D MoSIFT fuses the RGB data and depth information into the feature descriptors. First, Ming et al. (2012) adopt the same strategy using the RGB data to detect interest points. Then, for each interest point, 3D gradient space and 3D motion space are constructed by using RGB data and depth information. In 3D gradient (motion) space, they map 3D space into

three 2D planes: xy plane, yz plane and xz plane. Next, for each plane, they used SIFT algorithm to calculate the descriptors. Therefore, each 3D MoSIFT descriptor has 768 dimensions.

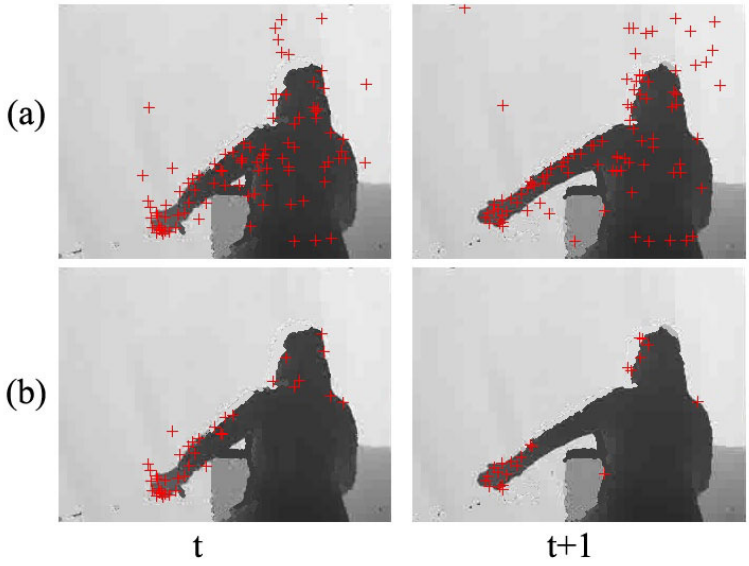

Figure 2: Results of interest point detection (marked with the red cross) in two consecutive frames. (a) 3D MoSIFT; (b) 3D EMoSIFT. We can see that some redundant points are detected in some slight motion regions (i.e., background regions) which shows 3D MoSIFT is sensitive to slight movement. However, 3D EMoSIFT can detect interest points from the regions with large motion (i.e., hand and arm regions), which shows 3D EMoSIFT is not sensitive to slight motion.

3. The Proposed Approach for One-Shot Learning Gesture Recognition

We propose a new spatio-temporal feature called 3D EMoSIFT. The new feature is invariant to scale and image-plane rotation. Then we use kmeans algorithm to learn codebook and apply SOMP algorithm to achieve descriptor coding. Besides, we adopt a methodology based on DTW and motion energy for temporal segmentation. Below, we describe each stage in detail.

3.1 Spatio-Temporal Feature Extraction: 3D EMoSIFT

The first stage is to extract rich spatio-temporal representations from the video clips. To obtain such representations, there are many ways to select (Dollár et al., 2005; Laptev, 2005; Chen and Hauptmann, 2009). However, those approaches only rely on RGB data and do not consider the depth information, which may lead to acquire insufficient information. Although 3D MoSIFT can fuse the RGB-D data to calculate descriptors, it still cannot accurately detect interest points. For instance, as shown in Figure 2(a), 3D MoSIFT capture some redundant interest points when some slight motion happens (e.g., slight motion in the background), showing that 3D MoSIFT is sensitive to slight movement. Besides, 3D MoSIFT (Ming et al., 2012) is a little sketchy. To solve the

mentioned problems, we propose a new spatio-temporal feature and give examples to explain how to extract the new feature step by step.

3.1.1 FEATURE POINTS DETECTION FROM RGB-D DATA

Although the 3D MoSIFT feature has achieved good results in human activity recognition, it still cannot eliminate some influences from the slight motion as shown in Figure 2(a). Therefore, we fuse depth information to detect robust interest points. We know that SIFT algorithm (Lowe, 2004) uses the Gaussian function as the scale-space kernel to produce a scale space of an input image. The whole scale space is divided into a sequence of octaves and each octave consists of a sequence of intervals, where each interval is a scaled image.

Building Gaussian Pyramid. Given a gesture sample including two videos (one for RGB video and the other for depth video),[1] a Gaussian pyramid for every grayscale frame (converted from RGB frame) and a depth Gaussian pyramid for every depth frame can be built via Equation (1).

$$\begin{aligned} L^I_{i,j}(x,y) &= G(x,y,k^j\sigma) * L^I_{i,0}(x,y),\ 0 \le i < n, 0 \le j < s+3, \\ L^D_{i,j}(x,y) &= G(x,y,k^j\sigma) * L^D_{i,0}(x,y),\ 0 \le i < n, 0 \le j < s+3, \end{aligned} \tag{1}$$

where (x,y) is the coordinate in an image; n is the number of octaves and s is the number of intervals; $L^I_{i,j}$ and $L^D_{i,j}$ denote the blurred image of the $(j+1)^{th}$ image in the $(i+1)^{th}$ octave; $L^I_{i,0}$ (or $L^D_{i,0}$) denotes the first grayscale (or depth) image in the $(i+1)^{th}$ octave; For $i=0$, $L^I_{0,0}$ (or $L^D_{0,0}$) is calculated from the original grayscale (depth) frame via bilinear interpolation and the size of $L^I_{0,0}$ is twice the size of the original frame; For $i>1$, $L^I_{i,0}$ (or $L^D_{i,0}$) is down-sampled from $L^I_{i-1,s}$ (or $L^D_{i-1,s}$) by taking every second pixel in each row and column. In Figure 3(a), the blue arrow shows that the first image $L^I_{1,0}$ in the second octave is down-sampled from the third image $L^I_{0,2}$ in the first octave. $*$ is the convolution operation; $G(x,y,k^j\sigma) = \frac{1}{2\pi(k^j\sigma)^2} e^{-(x^2+y^2)/(2(k^j\sigma)^2)}$ is a Gaussian function with variable-scale value; σ is the initial smoothing parameter in Gaussian function and $k = 2^{1/s}$ (Lowe, 2004). Then, the difference of Gaussian (DoG) images, Df, are calculated from the difference of two nearby scales in Equation (2).

$$Df_{i,j} = L^I_{i,j+1} - L^I_{i,j},\ 0 \le i < n, 0 \le j < s+2. \tag{2}$$

We give an example to intuitively understand the Gaussian pyramid and DoG pyramid. Figure 3 shows two Gaussian pyramids (L^{I_t}, $L^{I_{t+1}}$) built from two consecutive grayscale frames and two depth Gaussian pyramids (L^{D_t}, $L^{D_{t+1}}$) built from the corresponding depth frames. In this example, the number of octaves is $n=4$ and the number of intervals is $s=2$; Therefore, for each frame, we can build five images for each octave. And we can see that larger $k^j\sigma$ results in a more blurred image (see the enlarged portion of the red rectangle in Figure 3). Then, we use the Gaussian pyramid shown in Figure 3(a) to build the DoG pyramid via Equation (2), which is shown in Figure 4.

Building Optical Flow Pyramid. First, we briefly review the Lucas-Kanade method (Lucas et al., 1981) which is widely used in computer vision. The method assumes that the displacement of two consecutive frames is small and approximately constant within a neighborhood of the point ρ. The two consecutive frames are denoted by $F1$ and $F2$ at time t and $t+1$, respectively. Then

1. The depth values are normalized to [0 255] in depth videos.

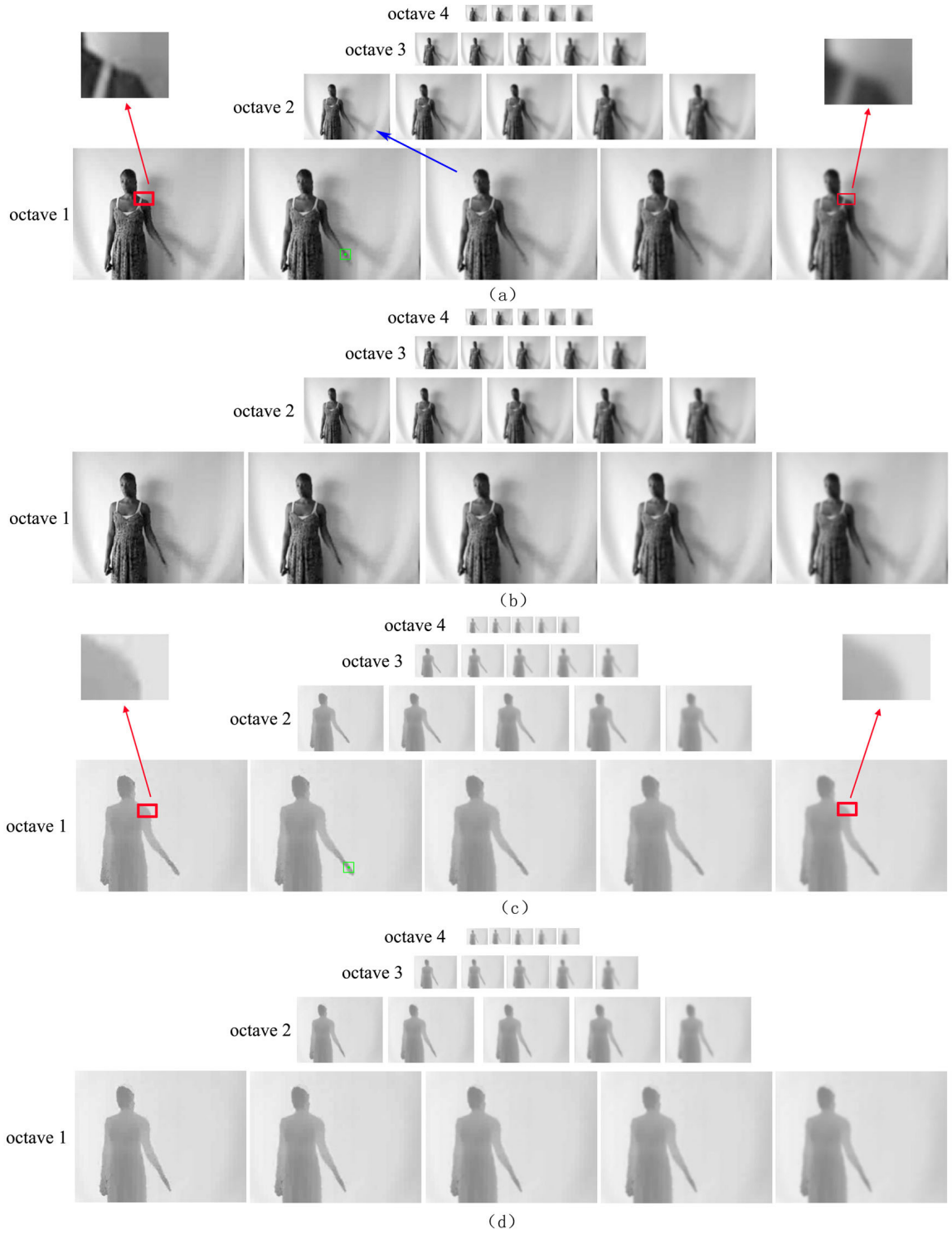

Figure 3: Building Gaussian pyramids and depth Gaussian pyramids for two consecutive frames. (a) the gaussian pyramid L^{I_t} at time t; (b) the gaussian pyramid $L^{I_{t+1}}$ at time $t+1$; (c) the depth gaussian pyramid L^{D_t} at time t; (d) the depth gaussian pyramid $L^{D_{t+1}}$ at time $t+1$.

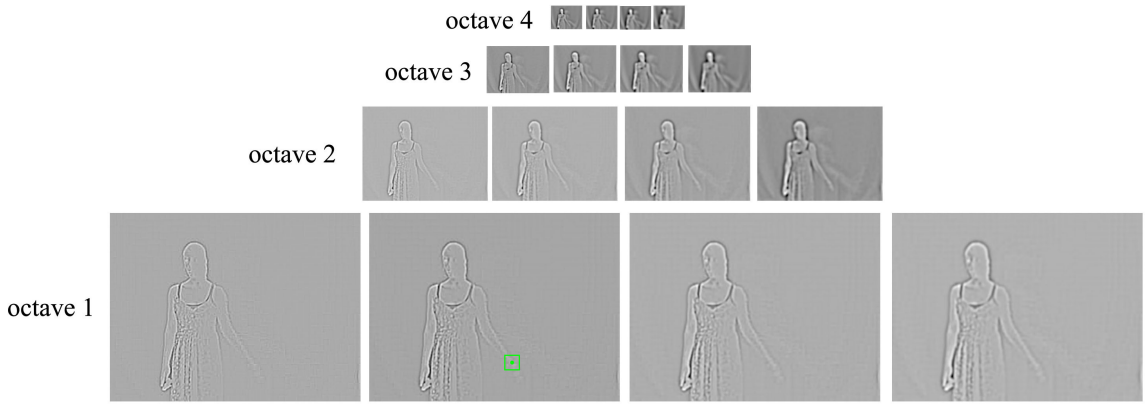

Figure 4: Building the difference of Gaussian pyramid Df^{I_t} from Figure 3(a) at time t.

the optical flow vector (v^{ρ}) of the point ρ can be solved by the least squares principle (Lucas et al., 1981). Namely, it solves:

$$Av^{\rho} = b,$$

where $A = \begin{bmatrix} F1_x(q_1) & F1_y(q_1) \\ F1_x(q_2) & F1_y(q_2) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ F1_x(q_n) & F1_y(q_n) \end{bmatrix}$, $v^{\rho} = \begin{bmatrix} v_x^{\rho} \\ v_y^{\rho} \end{bmatrix}$, and $b = \begin{bmatrix} -F1_t(q_1) \\ -F1_t(q_2) \\ \cdot \\ \cdot \\ \cdot \\ -F1_t(q_n) \end{bmatrix}$, $q_1, q_2, ..., q_n$ are the pixels inside the window around the point ρ, $F1_x(q_i)$ and $F1_y(q_i)$ calculated by different operators (e.g., Scharr operator, Sobel operator) are the partial derivatives of the image $F1$ along the horizontal and vertical directions, and $F1_t(q_i) = F2(q_i) - F1(q_i)$ calculated by two consecutive frames is the partial derivatives along time. Besides, v_x^{ρ} (v_y^{ρ}) denotes the horizontal (vertical) velocity of the point ρ. So we can know the optical flow $V = [V_x\ V_y]^T$ of all the points in the image $F1$ via Equation (3).

$$[V_x\ V_y]^T = \bigcup_{i=1}^{\zeta} [v_x^{\rho_i}\ v_y^{\rho_i}]^T, \tag{3}$$

where ζ is the number of points in the image $F1$, $v_x^{\rho_i}$ ($v_y^{\rho_i}$) denotes the horizontal (vertical) velocity of the point ρ_i, and V_x (V_y) denotes the horizontal (vertical) component of the estimated optical flow for all the points in an image. In order to facilitate the following description, we rewrite Equation (3), so as to define $OpticalFlowKL(F1, F2)$, as follow:

$$[V_x\ V_y]^T = OpticalFlowKL(F1, F2) \stackrel{def}{=} \bigcup_{i=1}^{\zeta} [v_x^{\rho_i}\ v_y^{\rho_i}]^T.$$

Next, once two Gaussian pyramids (L^{I_t} and $L^{I_{t+1}}$) shown in Figure 3(a) and (b) are obtained at time t and $t+1$, respectively, we can calculate the optical flow at each interval of each octave via Equation (4). That is say,

$$[V^{I_t}_{x,(i,j)}\ V^{I_t}_{y,(i,j)}]^T = OpticalFlowKL(L^{I_t}_{i,j}, L^{I_{t+1}}_{i,j}),\ 0 \le i < n, 0 \le j < s+3, \tag{4}$$

where $L^{I_t}_{i,j}$ denotes the blurred image of the $(j+1)^{th}$ interval in the $(i+1)^{th}$ octave at time t, n and s are defined the same as Equation (1).

So the horizontal and vertical optical flow pyramids at time t are the union sets $\bigcup_{i,j} V^{I_t}_{x,(i,j)}$ and $\bigcup_{i,j} V^{I_t}_{y,(i,j)}$, respectively. For example, we use the Gaussian pyramids in Figure 3(a) and (b) to compute the optical flow pyramid via Equation (4). And the results are illustrated in Figure 5(a) and (b) where we can see that the highlighted parts occur around the motion parts.

Local Extrema Detection. Here, we describe three different methods (SIFT, 3D MoSIFT, 3D EMoSIFT) for interest point detection and show the similarities and differences among these methods.

(1) Local Extrema Detection: SIFT

In order to detect the local maxima and minima in the DoG pyramid $Df^{I_t}_{i,j}$, each point is compared to its eight neighbors in the current image and nine neighbors in the above and below images of each octave, which is illustrated in Figure 6(a). A point is selected only if it is larger than all of these neighbors or smaller than all of them. In Figure 4, the DoG pyramid $Df^{I_t}_{i,j}$ has four octaves and each octave has four images at time t. So we can find the local extrema points in the middle of two images at each octave, namely, $Df^{I_t}_{i,j}$, $\forall i \in [0,3], j \in [1,2]$. For example, in the first octave, we detect the local extrema points at the second image $Df^{I_t}_{0,1}$ (via comparing the point to his 8 neighbor points in the current image $Df^{I_t}_{0,1}$, 9 neighbor points in the image $Df^{I_t}_{0,0}$, and 9 neighbor points in the image $Df^{I_t}_{0,2}$) and the third image $Df^{I_t}_{0,2}$ (via comparing the point to his 8 neighbor points in the current image $Df^{I_t}_{0,2}$, 9 neighbor points in the image $Df^{I_t}_{0,1}$, and 9 neighbor points in the image $Df^{I_t}_{0,3}$). So we can detect the local extrema points in other octaves similar to the first octave. The detected points (marked with red points) are shown in Figure 7(a), which shows that many redundant points are detected in the background and torso regions.

(2) Local Extrema Detection: 3D MoSIFT[2]

3D MoSIFT first detect the local extrema like SIFT algorithm. Then those local extrema can only become interest points when those points have sufficient motion in the optical flow pyramid. That is say, if a point is treated as an interest point, the velocity of this point should satisfy the following condition:

$$v_x \ge \beta_1 \times w, v_y \ge \beta_1 \times h, \tag{5}$$

where v_x (v_y) is the horizontal (vertical) velocity of a point from the horizontal (vertical) optical flow pyramid V_x (V_y); β_1 is a pre-defined threshold; w and h are the width and height of the blurred image in the scale space.

As shown in Figure 5(a) and (b), we can see that only the local extrema located in the highlighted parts of the optical flow pyramids ($V^{I_t}_x$ and $V^{I_t}_y$) will become interest points. Because only the points in the highlighted parts have large motions, which may satisfy the condition in Equation (5). Other extrema will be eliminated, because they have no sufficient motion in the optical flow pyramids. The final results (marked with red points) are shown in Figure 7(b).[3] Comparing with SIFT algorithm, we can see that if the points are still, they will be filtered out via the conditions in Equation (5).

2. MoSIFT and 3*D* MoSIFT have the same strategy to detect interest points.
3. Here, $\beta_1 = 0.005$ according to the reference (Ming et al., 2012).

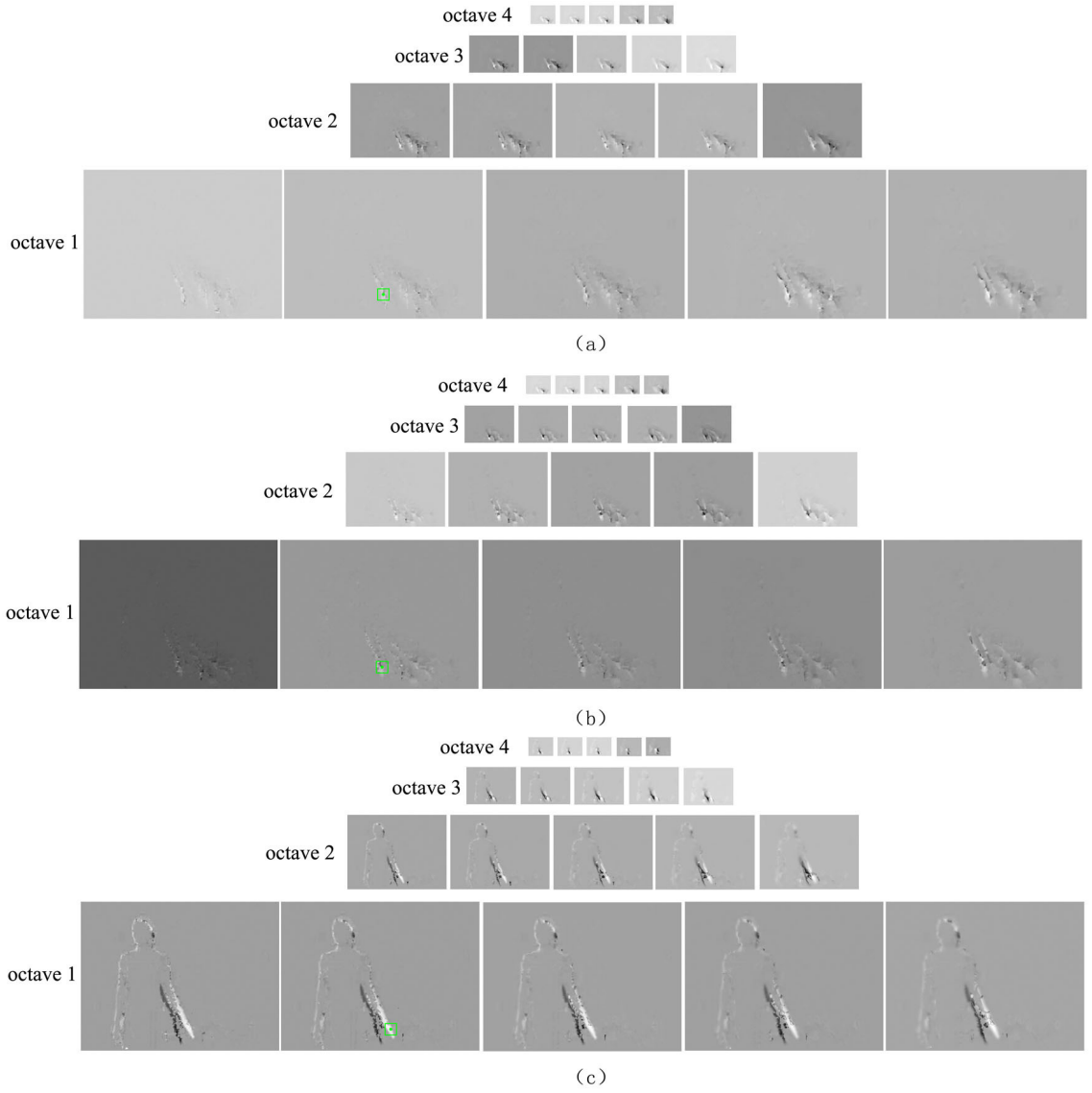

Figure 5: The horizontal and vertical optical flow pyramids are calculated from Figure 3(a) and (b). (a) The horizontal component of the estimated optical flow pyramid $V_x^{I_t}$ at time t; (b) The vertical component of the estimated optical flow pyramid $V_y^{I_t}$ at time t; (c) The depth changing component $V_z^{D_t}$ at time t.

However, in Figure 7(b), some useless points (from the background and torso regions) are still detected, which indicate that 3D MoSIFT is sensitive to the slight motion.

(3) Local Extrema Detection: 3D EMoSIFT

To eliminate the effect of the slight motion, we introduce a new condition to filter out the detected points by the SIFT algorithm. According to the above mentioned description, we have ob-

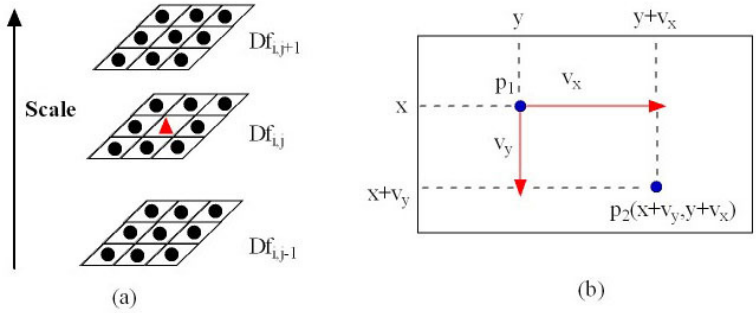

Figure 6: (a) The SIFT algorithm for interest points detection. Maxima and minima of the DoG images are detected by comparing a pixel (marked with a red triangle) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with black circles); (b) the point prediction via the optical flow vector.

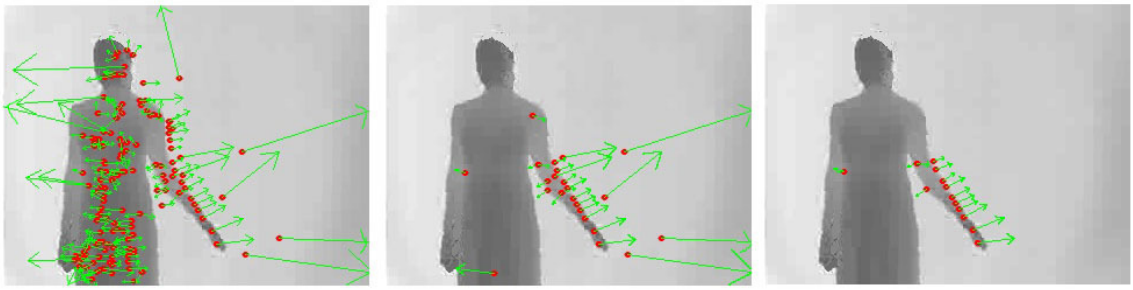

Figure 7: After interest point detection, the SIFT-based descriptors are calculated by three methods: SIFT, 3D MoSIFT and 3D EMoSIFT. The detected points are marked with red circles and the green arrows show the direction of movements. The figure shows that SIFT and 3D MoSIFT detect many useless points in the background and torso regions while the result by 3D EMoSIFT is more accurate. (a) SIFT; (b) 3D MoSIFT; (c) 3D EMoSIFT.

tained the pyramids $L^{D_t}, L^{D_{t+1}}, V_x^{I_t}, V_y^{I_t}$. For a given point p_1 from an image in different scale spaces at time t, we can easily know the horizontal and vertical velocities v_x, v_y by the corresponding image of the pyramids $V_x^{I_t}, V_y^{I_t}$. Then the predicted point p_2 at time $t+1$ can be calculated by the point p_1 at time t according to Figure 6(b). Therefore, we can know the depth changing component at time t as:

$$V_{z,(i,j)}^{D_t}(p_1) = L_{i,j}^{D_{t+1}}(p_2) - L_{i,j}^{D_t}(p_1),\ 0 \le i < n, 0 \le j < s+3. \tag{6}$$

Figure 5(c) shows the depth changing pyramid via Equation (6). We can see that the highlighted parts accurately occur in the gesture motion region. Therefore, the local extrema shown in Figure 7(a) by SIFT algorithm will become interest points when those points not only have sufficient motion which is satisfied with the condition of 3D MoSIFT in Equation (5) but also have enough depth

changing which is shown in the highlighted regions of Figure 5(c). That is say, the interest point detection must simultaneously satisfy the condition in Equation (5) and a new condition defined as:

$$v_z \geq \beta_2 \times \sqrt{w^2 + h^2}, \tag{7}$$

where v_z is the depth changing value of a point from the depth changing pyramid V_z; β_2 is a pre-defined threshold. The final results is shown in Figure 7(c).[4] We can see that 3D EMoSIFT can filter out the still points and the points with slight motion.

3.1.2 FEATURE DESCRIPTORS

The previous operations assigned an image location and scale to each interest point. That is say we can use the interest point to select the Gaussian images from different pyramids. Here, we give an example to illustrate how to compute the feature descriptor vector which is similar to the process in Ming et al. (2012). We assume that a detected point (marked with green dot) is found in DoG pyramid $Df_{0,1}^{I_t}$ at time t in Figure 4, which indicates that the detected point locates at the second image of the first octave. Then the corresponding points (marked with green dot) in different pyramids are shown in Figure 3 and Figure 5 at time t. To calculate the feature descriptors, we first extract the local patches ($\Gamma_1 \sim \Gamma_5$) around the detected point in five pyramids ($L^{I_t}, L^{D_t}, V_x^{I_t}, V_y^{I_t}$ and $V_z^{I_t}$), where Γ_1 is extracted from $L_{0,1}^{I_t}$, Γ_2 from $L_{0,1}^{D_t}$, Γ_3 from $V_{x,(0,1)}^{I_t}$, Γ_4 from $V_{y,(0,1)}^{I_t}$ and Γ_5 from $V_{z,(0,1)}^{D_t}$. These five patches are labeled as green rectangles in Figure 3 and Figure 5. The local patches $\Gamma_1 \sim \Gamma_5$ are of the same size 16×16 pixels and are shown in Figure 8. We first consider the appearance properties to construct the 3D gradient space via local patches Γ_1 and Γ_2. Then we use the rest of local patches (Γ_3, Γ_4 and Γ_5) to construct 3D motion space.

Feature Descriptors in 3D Gradient Space. For a given point p with its coordinate (i, j), we can simply calculate its horizontal and vertical gradients from RGB-D data (Γ_1 and Γ_2) as follow:

$$\begin{aligned} I_x(i,j) &= \Gamma_1(i, j+1) - \Gamma_1(i,j), \\ I_y(i,j) &= \Gamma_1(i+1, j) - \Gamma_1(i,j), \\ D_z^x(i,j) &= \Gamma_2(i, j+1) - \Gamma_2(i,j), \\ D_z^y(i,j) &= \Gamma_2(i+1, j) - \Gamma_2(i,j), \end{aligned}$$

where $I_x(i,j)$ and $I_y(i,j)$ are the horizontal and vertical gradients calculated from Γ_1; D_z^x and $D_z^y(i,j)$ are the horizontal and vertical gradients from Γ_2. We can calculate four gradients (I_x, I_y, D_z^x and D_z^y) for each point. Because the local patches (Γ_1 and Γ_2) are of size 16×16, there are 256 points and each point has four gradient values.

Then, as shown in Figure 8(a), for each point p, the 3D gradient space can be constructed by $I_x(i,j), I_y(i,j), D_z^x(i,j)$ and $D_z^y(i,j)$. Now we use the xy plane to illustrate how to calculate the feature descriptor in the 3D gradient space. For each point p with its coordinate (i, j), we compute the gradient magnitude, $mag(i,j) = \sqrt{I_x(i,j)^2 + I_y(i,j)^2}$, and orientation, $ori(i,j) = tan^{-1}(I_y(i,j)/I_x(i,j))$ in the xy plane. Then, in xy plane, we can generate a new patch Γ_{xy} which is the left image in the first row of Figure 8(c). The size of Γ_{xy} is 16×16. For each point with its coordinate (i, j) from Γ_{xy}, it has two values: the gradient magnitude $mag(i,j)$ and orientation $ori(i,j)$. Γ_{xy} can be divided into

4. Here, $\beta_1 = \beta_2 = 0.005$.

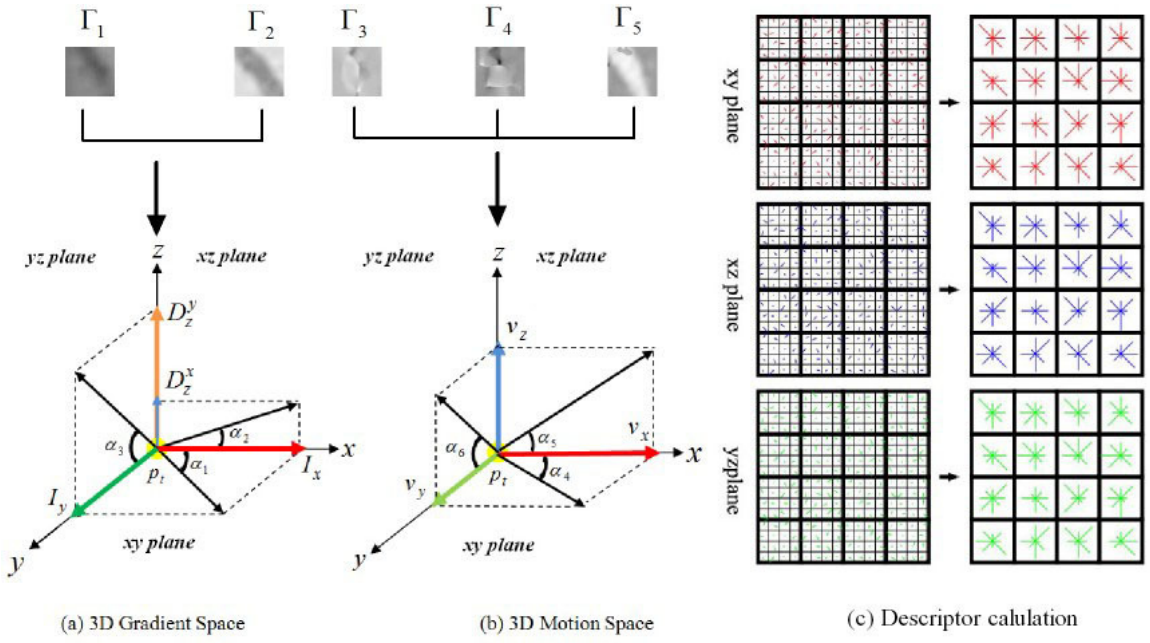

Figure 8: Computing the feature descriptor in two parts: (a) 3D Gradient Space, (b) 3D Motion Space, (c) Feature descriptor calculation

16 (4×4) grids. For each grid with 4×4 points, we calculate its orientation histogram with 8 bins, which means the orientation is grouped into 8 directions which is represented by the right image in the first row of Figure 8(c). This leads to a descriptor vector with 128 ($4 \times 4 \times 8$) dimensions in xy plane. Here, each sample added to the histogram is weighed by its gradient magnitude and by a Gaussian weighting function (Lowe, 2004). Similarly, we can calculate the descriptors in xz and yz planes. Therefore, the descriptor vector of the 3D gradient space has 384 (128×3) dimensions.

Feature Descriptors in 3D Motion Space. For a given point p with coordinates $(i, j), \forall\ 0 \leq i \leq 15,\ 0 \leq j \leq 15$, we can easily know the velocities according to the local patches Γ_3, Γ_4, and Γ_5. That is say, $v_x(i, j) = \Gamma_3(i, j)$, $v_y(i, j) = \Gamma_4(i, j)$ and $v_z(i, j) = \Gamma_5(i, j)$.

Thus, we can construct the 3D motion space as shown in Figure 8(b). Similar to the descriptor calculation in 3D gradient space, we can compute the magnitude and orientation (using v_x, v_y, v_z) for the local patch around the detected points in three planes. The only difference is that v_z is the same in both xz and yz planes. Therefore, we obtain the descriptors with 384 dimensions in the 3D motion space. Finally, we integrate these two descriptor vectors into a long descriptor vector with 768 dimensions.

3.1.3 OVERVIEW THE 3D EMOSIFT FEATURES

In this section, we propose a new spatio-temporal feature called 3D EMoSIFT. Each 3D EMoSIFT feature descriptor has 768 dimensions. Since the 3D EMoSIFT feature is derived from SIFT algorithm, the features are invariant to scale and rotation. Besides, compared to other similar features (SIFT, MoSIFT, 3D MoSIFT), the new features can capture more compact motion patterns and are

not sensitive to the slight motion (see the Figure 7). For a given sample including an RGB video and a depth video, we can calculate feature descriptors between two consecutive frames. Then the sample can be represented by the set of all the feature descriptors extracted from the video clips. Algorithm 1 illustrates how to calculate the proposed features.

Now each sample is denoted by the set of descriptor vectors, and we want to use those vectors for BoF representation. To do that, we will create histograms counting how many times a descriptor vector (representing a feature) appears at interest points anywhere in the video clip representing the gesture. There is a need to first replace the descriptor vectors by codes to limit the number of features, otherwise there would be too many entries in the histogram and the representation would be too sparse. So, we will describe the means of creating a codebook in the next Section 3.2.

Algorithm 1 The algorithm for the 3D EMoSIFT feature

Input:

- A sample with two videos: $V_r = [I_1, I_2, ..., I_Q]$ (RGB data), $V_d = [D_1, D_2, ..., D_Q]$ (depth data)
- Number of frames : Q

Output:

- The set of feature descriptors : X

Initialization: $X = [\,]$
for $i = 1$ to $Q-1$ **do**
  Obtain the frames: I_i and I_{i+1} from V_r; D_i and D_{i+1} from V_d
  Build the Gaussian Pyramids: $L^{I_i}, L^{I_{i+1}}, L^{D_i}$ and $L^{D_{i+1}}$ via Equation (1)
  Build the different of Gaussian (DoG) Pyramid: Df^{I_i} via Equation (2)
  Build the Optical Flow Pyramids: $V_x^{I_i}$ and $V_y^{I_i}$ via Equation (4)
  Build the depth changing Pyramid: V_z^{Di} via Equation (6)
  Find the set of interest points: $P = [p_1, ..., p_m]$ via Figure 6(a), Equation (5) and (7)
  for $j = 1$ to m **do**
    Get the information of the interest point from the set P: p_i
    Compute feature descriptor from the local patch around p_i: $x \in \Re^{768}$ via Figure 8
    $X = [X\ x]$
  end for
end for
return X

3.2 Codebook Learning and Coding Descriptors

Suppose the matrix X is the set of all descriptor vectors for an entire video clip representing a gesture, and $X = [x_1, x_2, ..., x_N] \in \Re^{d \times N}$, where x_i denotes a description with d dimensions. A codebook B with M entries is denoted with $B = [b_1, b_2, ..., b_M] \in \Re^{d \times M}$. The coding methods map each descriptor into a M-dimensional code to generate the video representation. We first introduce how to learn a codebook B, then review VQ and introduce SOMP for code descriptors.

3.2.1 CODEBOOK LEARNING

Let η denote the number of gesture classes (that means there are η training samples for one-shot learning), $\Omega = [X^1, X^2, ..., X^\eta]$, $\Omega \in \Re^{d \times L_{tr}}$ is the set of all the descriptor vectors extracted from all the training samples, $X^i \in \Re^{d \times N_i}$ with N_i descriptor vectors is the set extracted from the i^{th} class, and $L_{tr} = \sum_{i=1}^{\eta} N_i$ is the number of features extracted from all the training samples. Then we learn the codebook $B \in \Re^{d \times M}$ ($M < \sum_{i=1}^{\eta} N_i$) with M entries by applying the k-means algorithm (Wang et al., 2010) over all the descriptors Ω in our work. However, unlike traditional BoF models, we use a new parameter $\gamma \in (0,1)$ instead of the codebook size M (The way we select γ will be discussed in Section 4.). γ is expressed as a fraction of L_{tr}. Therefore, the codebook size M can be calculated below:

$$M = L_{tr} \times \gamma. \tag{8}$$

3.2.2 CODING DESCRIPTORS BY VQ

In the traditional VQ method, we can calculate the Euclidean distance between a given descriptor $x \in \Re^d$ and every codeword $b_i \in \Re^d$ of the codebook B and find the closest codeword. The VQ method can be formulated as:

$$\min_C \|X - BC\|_F^2, \ s.t. \|c_i\|_0 = 1, \|c_i\|_1 = 1, c_i \geqslant 0, \ \forall \ i, \tag{9}$$

where $\|\cdot\|_F$ is the Frobenius norm, $C = [c_1, c_2, ..., c_N] \in \Re^{M \times N}$ is the set of codes for X, $\|\cdot\|_0$ is the ℓ_0 norm that counts the number of nonzero elements, $\|\cdot\|_1$ is the ℓ_1 norm; The conditions $\|c_i\|_0 = 1, \|c_i\|_1 = 1, c_i \geqslant 0$, mean that only one element is equal to 1 and the others are zero in each code $c_i \in \Re^M$.

This formulation in Equation (9) allows us to compare more easily with sparse coding (see the Section 3.2.3). In Equation (9), the conditions may be too restrictive, which gives rise to usually a coarse reconstruction of X. Therefore, we use a sparse coding method instead of VQ.

3.2.3 CODING DESCRIPTORS BY SOMP

Inspired by image classification (Yang et al., 2009) and robust face recognition (Wright et al., 2009) via sparse coding, we relax the restricted conditions in Equation (9) and suppose X has a sparse representation $C = [c_1, c_2, ..., c_N]$, $c_i \in \Re^M$ that means each c_i contains k ($k \ll M$) or fewer nonzero elements. Then, the problem can be stated as the following optimization problem:

$$\min_C \|X - BC\|_F^2, \ s.t. \|c_i\|_0 \leq k, \ \forall \ i. \tag{10}$$

Solving Equation (10) accurately is an NP-hard problem (Wright et al., 2009; Guo et al., 2013). Nevertheless, approximate solutions are provided by greedy algorithms or convex relaxation, such as SOMP (Tropp et al., 2006; Rakotomamonjy, 2011). To the best of our knowledge, we are the first to use SOMP in BoF model for gesture recognition, especially for one-shot learning gesture recognition.

Then we give a brief introduction about the SOMP algorithm and analyze the computational complexity. SOMP is a greedy algorithm which is based on the idea of selecting an element of the codebook and building all signal approximations as the projection of the signal matrix X on the span

of these selected codewords. This algorithm (Tropp et al., 2006; Rakotomamonjy, 2011) is shown in Algorithm 2. Regarding the computational complexity, we note that the most demanding part of the SOMP is the correlation E computation which has the complexity $O(dMN)$. And the complexity of the linear system to be solved for obtaining C at each iteration is $O(|\Lambda|)$. So the complexity for k iterations is about $O(dkMN) + O(k|\Lambda|)$. Although the complexity of SOMP is more expensive than VQ which has $O(dMN)$ (Linde et al., 1980). SOMP has several merits which will be discussed later.

Algorithm 2 The SOMP algorithm

Input:

- A signal matrix (the feature set): $X = [x_1, x_2, ..., x_N] \in \Re^{d \times N}$
- A learned codebook: $B = [b_1, b_2, ..., b_M] \in \Re^{d \times M}$
- the sparsity: k

Output:

- The sparse representation: C

Initialization: the residual matrix $R_s = X$, the index set $\Lambda = [\,]$;
for $i = 1$ to k **do**
  $E = B^T R_s$, where $E = \cup_{p,q}[e_{p,q}]$
  Find the index $\lambda = argmax_q \sum_p |e_{p,q}|$
  $\Lambda = [\Lambda\ \lambda]$
  $C = (B_\Lambda^T B_\Lambda)^{-1} B_\Lambda^T X$
  $R_s = X - BC$
end for
return C

When the codebook $B \in \Re^{d \times M}$ and a descriptor set $X \in \Re^{d \times N}$ are given, the set of codes $C \in \Re^{M \times N}$ can be calculated by the coding methods (VQ or SOMP). Then the mean reconstruction error (MRE) for X is defined as:

$$\varepsilon_{MRE} = \sum_{i=1}^{N} \varepsilon_i / N,$$

where $\varepsilon_i = \|x_i - Bc_i\|_2^2$ is the reconstruction error of the i^{th} descriptor.

To compare the $MREs$ for both the VQ and SOMP methods, a matrix $X \in \Re^{64 \times 2000}$ is randomly generated based on the standard normal distribution. Then the matrix X is split into two parts ($X_1 \in \Re^{64 \times 1000}$ and $X_2 \in \Re^{64 \times 1000}$). The matrix X_1 is used to build a codebook B by the k-means algorithm. Then we use X_2 to calculate the codes C_{VQ} and C_{SOMP} via Equation (9) and (10), respectively. Finally we calculate the $MREs$ under varied cluster numbers and different sparsity values $k = \{5, 10, 15\}$. Figure 9 shows the results of both coding methods. We can see that the $MREs$ of the SOMP method is much lower than the $MREs$ of the VQ method.

Compared with the VQ method, SOMP has several advantages. First, the codebook B is usually overcomplete (i.e., $M > d$). Overcomplete codings smoothly interpolate between input vectors and are robust under input noise (Olshausen et al., 1997). Second, SOMP achieves a much lower reconstruction error. Although there is no direct relationship between lower reconstruction error and good recognition results, some authors (Yang et al., 2009; Wan et al., 2012) have shown that oftentimes better reconstruction leads to better performance. Third, the sparsity prior allows the

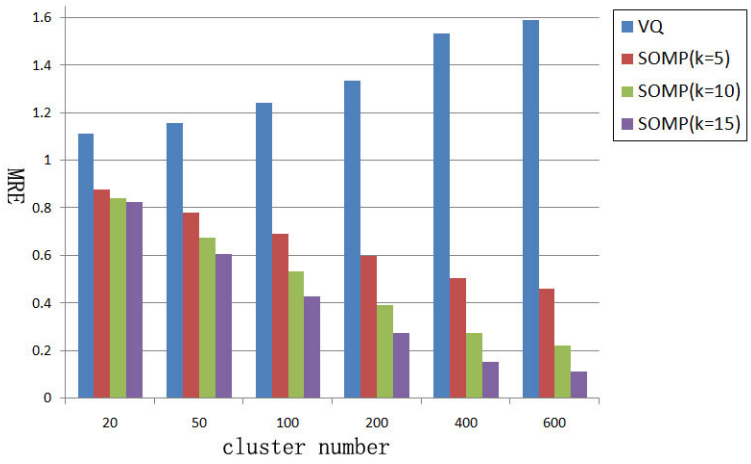

Figure 9: Comparison *MREs* using both VQ and SOMP methods.

learned representation to capture salient patterns of local descriptors. According to our experimental results in Section 4, although VQ can produce satisfactory accuracy, SOMP can achieve better performance.

3.3 Coefficient Histogram Calculation and Classification

The matrix X contains the descriptors obtained from a test sample and C contains their corresponding sparse representations over the learned codebook B. The sparse coefficients of the vector $c_i \in C$ present the contribution of all the entries in approximating the descriptor $x_i \in X$. The sparse coefficients associated with all the descriptors of the test sample thus collectively demonstrate the contribution of the entries toward the representation of that sample. Therefore, we use the coefficient histogram to denote the representation of each individual sample via Equation (11).

$$h = \frac{1}{N}\sum_{i=1}^{N} c_i, \tag{11}$$

where $c_i \in \Re^M$ is the i^{th} descriptor of $C \in \Re^{M\times N}$, and N is the total number of descriptors extracted from a sample and $h \in \Re^M$.

Because we have only one sample per class for training, multi-class SVMs are not trivially applicable because they require in principle a large number of training examples. So we select the NN classification for gesture recognition.

In the above discussion, we assume that every video has one gesture but this assumption is not suitable for continuous gesture recognition system. Therefore, we first apply DTW to achieve temporal gesture segmentation, which splits the multiple gestures to be recognized. We use the sample code about DTW provided in ChaLearn gesture challenge website (`http://gesture.chalearn.org/data/sample-code`). The detailed description of how to use DTW in one-shot learning can be found in Guyon et al. (2013). We briefly introduce the process for temporal gesture segmentation by DTW so as to make this paper more self-contained.

3.4 Temporal Gesture Segmentation based on DTW

Let $V = [I_1, ..., I_N]$ be a video with N frames, where I_i is the i^{th} frame (grayscale image) in the video. A video is represented by a set of motion features obtained from difference images as follows. First, the difference image is computed by subtracting consecutive frames in a video, that is $E_i = I_{i+1} - I_i$, $i = 1, ..., N-1$. The difference image is shown in Figure 10(b). Then a grid of equally spaced cells is defined over the difference image. The default size of the grid is 3×3 as shown in Figure 10(c). For each cell, we calculate the average value in the difference image, so a 3×3 matrix is generated. Finally, we flatten this matrix into a vector which is called motion feature. Therefore, a video V with N frames is represented by a matrix (the set of motion features) $f_V \in \Re^{9 \times (N-1)}$.

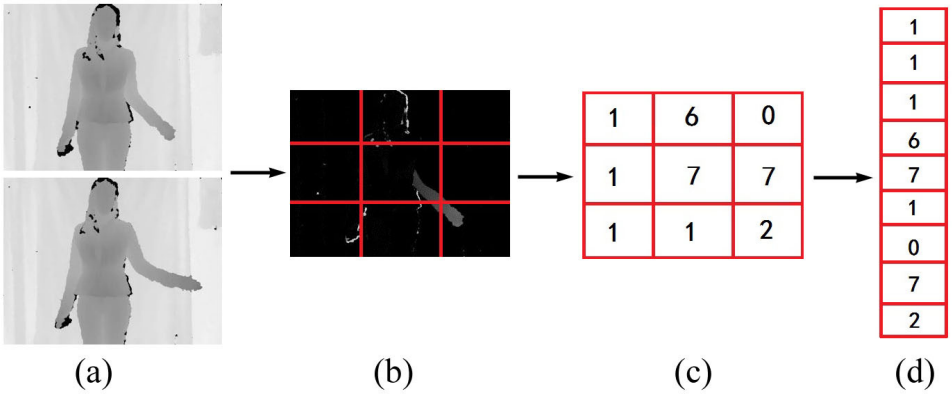

Figure 10: An example for the calculation of motion feature vector.

The reference sequence with κ training videos is denoted by $F_{tr} = [f_{Vtr_1}, ..., f_{Vtr_\kappa}]$, f_{Vtr} is the set of motion features of a training video. A test sequence is denoted by $F_{te} = f_{Vte}$ (the set of motion features for the test video). We calculate the negative Euclidean distance between each entry (a motion feature) from F_{tr} and each entry (a motion feature) from F_{te}. Then we calculate the DTW distance and apply the Viterbi algorithm (Viterbi, 1967) to find the temporal segmentation (an optimal warping path). In Figure 11, the left gray image shows the set of motion features (F_{tr}) as the reference sequence calculated from training videos. A motion feature (F_{te}) as the test sequence is computed from a new input video. The optimal path is shown in the top right corner (the green line is the optimal path; the short red lines are the boundary of two neighboring gestures). We can see that the testing video is splitted into five gestures.

3.5 Overview of the Proposed Approach

In this section, we describe the proposed approach based on bag of 3D EMoSIFT features for one-shot learning gesture recognition in detail. In the recognition stage, it has five steps: temporal gesture segmentation by DTW, feature descriptor extraction using 3D EMoSIFT, coding descriptor via SOMP, coefficient histogram calculation and the recognition results via NN classifier. The overall process is summarized in Algorithm 3.

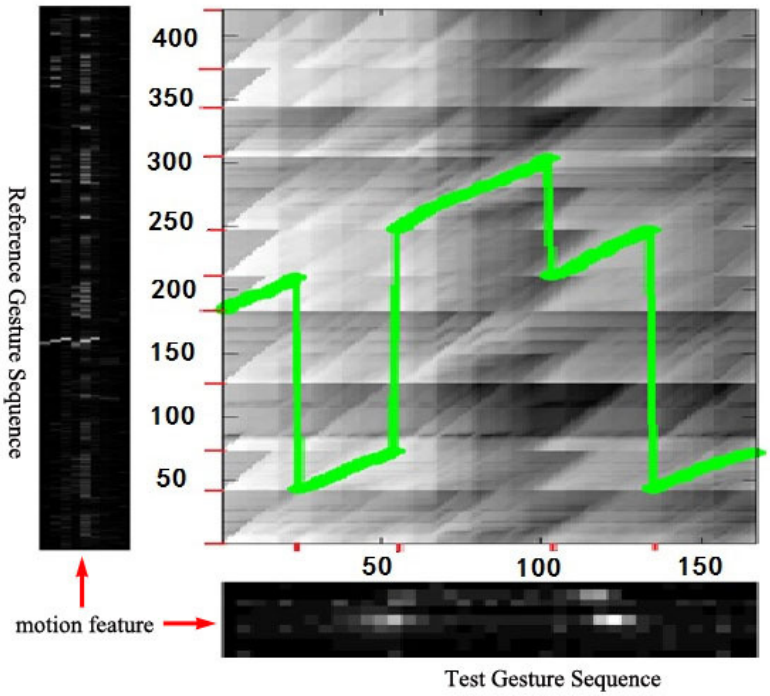

Figure 11: Temporal gesture segmentation by DTW.

4. Experimental Results

This section summarizes our results and demonstrates the proposed method is well suitable for one-shot learning gesture recognition. We first discuss the parameters of the proposed method. We further extend our method to compare with other state-of-the-art methods. Our experiments reveal that the proposed method gives superior recognition performance than many existing approaches.

4.1 Database

We evaluate the proposed method on development batches ($devel01 \sim devel20$), validation batches ($valid01 \sim valid20$) and final batches ($final21 \sim final40$) which contain in total 6000 gestures. The sixty batches are from Chalearn gesture challenge. Each batch is made of 47 gesture videos and split into a training set and a test set. The training set includes a small set of vocabulary spanning from 8 to 15 gestures. Every test video contains 1 to 5 gestures. Detailed descriptions of the gesture data can be found in Guyon et al. (2012). All the samples are recorded with a Microsoft $Kinect^{TM}$ camera which provides both RGB and depth video clips. Some examples are shown in Figure 12 where the first row is RGB images and the corresponding depth images are shown in the second row.

4.2 Metric of Evaluation

We adopt the metric of evaluation that was used by the challenge organizers (Guyon et al., 2012) to rank the entries. To evaluate performance, we use Levenshtein distance to calculate the score between the predicted labels and the truth labels. This distance between two strings is defined as

Algorithm 3 The proposed approach for one-shot learning gesture recognition

The condition for one-shot learning: given K training samples (RGB-D data) for K class (one sample per gesture class).

Input:

- Training samples (RGB-D data): $T_r = [t_{r1}, ..., t_{rK}]$
- A learned codebook: B (computed from training stage)
- Coefficient histograms of training samples: $H_r = [h_{r1}, h_{r2}, ..., h_{rK}]$ via Equation (11) (computed from training stage)
- A test sample (RGB-D data): t_e

Output:

- The recognition results: $class$

Initialization: $class = [\,]$
Temporal gesture segmentation: $[t_{e_1}, t_{e_2}, ..., t_{e_N}] = DTW(T_r, t_e), N \geq 1$
for $i = 1$ to N **do**
  Spatio-temporal feature extraction: $X_{t_e} = 3D\_EMoSIFT(t_{e_i})$
  For X_{t_e}, calculate its sparse representation C over the pre-trained codebook B
  $\min_C \|X_{t_e} - BC\|_F^2 \quad s.t. \quad \|c_j\|_0 \leq k, \forall\ j$
  Calculate the coefficient histogram h_{t_e} via Equation (11)
  Recognition: $tmp\_calss = nn\_classify(H_r, h_{t_e})$
  $class = [class\ tmp\_calss]$
end for
return $class$

Figure 12: Some samples from ChaLearn gesture database.

the minimum number of operations (insertions, substitutions or deletions) needed to transform one string into the other. In our case, the strings contain the gesture labels detected in each sample. For all comparisons, we compute the mean Levenshtein distance (MLD) over all video clips and batches. MLD score is analogous to an error rate (although it can exceed 1).

4.3 Parameters Discussion

This part gives the discussion of the parameters of the proposed method. First, we analysis the parameters of 3D EMoSIFT. Then, two parameters from the BoF model are discussed.

4.3.1 PARAMETERS OF 3D EMOSIFT

There are five parameters for constructing 3D EMoSIFT features. Three parameters σ, n and s in Equation (1) are derived from SIFT algorithm. We set $\sigma = 1.6$ and $s = 3$. Because Lowe (2004) suggest that when $\sigma = 1.6$ and $s = 3$, they can provide the optimal repeatability according to their experimental results. Besides, the number of octaves n can be calculated according to the original image size, such as $int(log_2(min(width, height)))$ (Vedaldi and Fulkerson, 2008).

The rest of parameters are β_1 in Equation (5) and β_2 in Equation (7). β_1 and β_2 determine the detection of interest points based on motion and depth change. When β_1 and β_2 are smaller, more interest points will be detected. We find that when $\beta_1 \in [0.003\ 0.008], \beta_2 \in [0.003\ 0.008]$, the performances are very stable as shown in Figure 13 where the results are calculated from two batches. We can see that MLD scores vary from 0.075 to 0.092 for *devel*01 batch, from 0.089 to 0.134 for *devel*02 batch. Therefore, $\beta_1 = \beta_2 = 0.005$ is used throughout this paper based on empirical results.

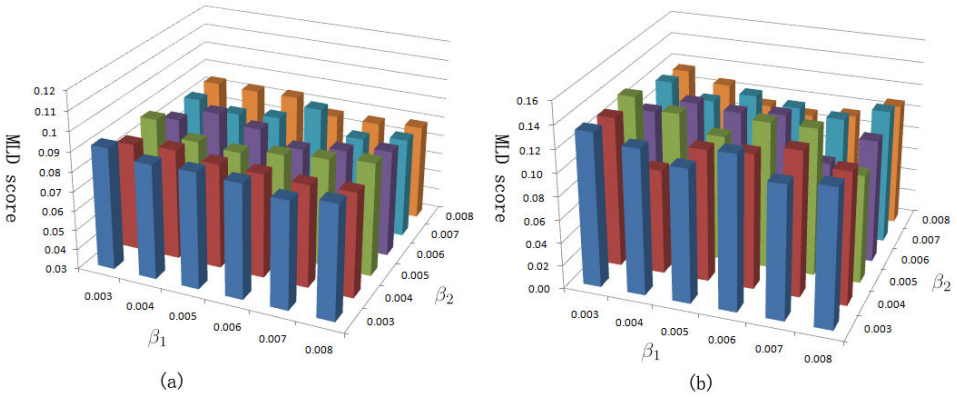

Figure 13: Parameters: $\sigma = 1.6$, $s = 3$, $\gamma = 0.2$ and $k = 10$. The MLD scores are calculated with different values β_1, β_2. (a) on *devel*01 batch (b) on *devel*02 batch

4.3.2 PARAMETERS OF THE BOF MODEL

There are two parameters in the BoF model: γ in Equation (8) and k in Equation (10) . Unlike traditional BoF models, we use a new parameter $\gamma \in (0, 1)$ to replace the codebook size M mentioned in Section 3.2. We first explain the reasons for choosing γ. Table 2 shows some information on different batches ($final21 \sim final40$), such as the number of training samples and the number of features extracted from training samples. We can see that the number of features varies on different batches. If a given codebook size M is too large, it may cause over-clustering on some batches where the number of features is relatively fewer (e.g., $final25$ and $final36$). Therefore, the over-clustering will effect the final MLD score. For instance, we evaluate seven different codebook sizes: {800, 1000, 1500, 2000, 2500, 3000, 3500}. The corresponding results are shown in Table 3 where the best performance is 0.18242. Then we evaluate different values {0.1, 0.2, 0.3} for γ, and the results are shown in Table 4. We can see that even though $\gamma = 0.1$, the corresponding MLD score is 0.17415 which can easily beat the best performance in Table 3. Additionally, when $\gamma = 0.1$, the

corresponding mean codebook size 1440 is much smaller than the given codebook size 3500 which is from the best result in Table 3.

The theory of sparse coding and the codebook learning are in a developing stage and the problems for selecting optimal parameters (e.g., γ, sparsity k) are still open issues (Guha and Ward, 2012). In this paper, we use a simple strategy to decide these two parameters. At first, we keep $k = 10$ and set γ with different values (ranging from 0.1 to 0.5), then determine γ by the lowest MLD score. Figure 14(a) shows the results. It reveals when $\gamma = 0.5$, we can get a higher performance and the corresponding MLD score is 0.13145. Then we set different values of k with $\gamma = 0.5$ and the results are shown in Figure 14(b). We can see that MLD scores remain stable. When $\gamma = 0.5$ and $k = 12$, the proposed method gets the lowest MLD score (the corresponding value is 0.1259).

batch names	number of training samples:N_{tr}	number of features (3D MoSIFT):$L1_{tr}$	number of features (3D EMoSIFT):$L2_{tr}$	decrease in ratio:$1-\frac{L2_{tr}}{L1_{tr}}$
final21	10	18116	13183	27.23%
final22	11	19034	15957	16.17%
final23	12	11168	7900	29.26%
final24	9	10544	7147	32.22%
final25	11	8547	6180	27.69%
final26	9	9852	7675	22.10%
final27	10	29999	20606	31.31%
final28	11	16156	10947	32.24%
final29	8	30782	22692	26.28%
final30	10	20357	14580	28.38%
final31	12	22149	17091	22.84%
final32	9	12717	10817	14.94%
final33	9	42273	29034	31.32%
final34	8	24099	16011	33.56%
final35	8	39409	27013	31.45%
final36	9	9206	6914	24.90%
final37	8	22142	14181	35.95%
final38	11	26160	18785	28.19%
final39	10	16543	11322	31.56%
final40	12	11800	10128	14.17%
Average	9.85	20052.65	14408.15	**28.15%**

Table 2: This table shows some information for every batch. The last row reveals the average number. Although the average number of 3D EMoSIFT features has decreased by 28.15%, 3D EMoSIFT has a higher performance than 3D MoSIFT in our experimental results. Besides, compared 3D MoSIFT features, the process time of 3D EMoSIFT can be faster to build the cookbook.

4.4 Comparisons

In order to compare with other methods, we first use the standard BoF model to evaluate different spatio-temporal features. Then the performances of VQ and SOMP is given. Besides, we evaluate

codebook size M	800	1000	1500	2000	2500	3000	3500
MLD score	0.21448	0.21504	0.19514	0.18961	0.18684	0.18574	**0.18242**

Table 3: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$ and $k = 10$ ($final21 \sim final40$). MLD scores with different codebook sizes M.

γ	0.1	0.2	0.3
MLD score	0.17415	0.14753	0.14032
Mean codebook size	1440	2881	4322

Table 4: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$ and $k = 10$ ($final21 \sim final40$). MLD scores with different values for γ.

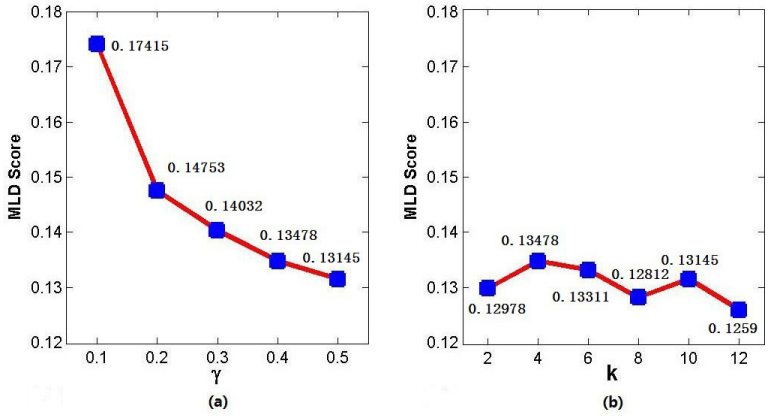

Figure 14: (a) Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$ and $k = 10$ ($final21 \sim final40$). MLD scores with different values of γ; (b) Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$ and $\gamma = 0.5$ ($final21 \sim final40$). MLD scores with different values of sparsity k.

the performances of both the gradient-based and motion-based features. Finally, we compare the proposed approach with some popular sequence matching methods.

4.4.1 COMPARISON WITH OTHER SPATIO-TEMPORAL FEATURES

In our experiments, we use the standard BoF model to evaluate different spatio-temporal features, which means VQ is used for coding descriptors. As shown in Figure 14(b), the results are relatively stable when sparsity k has different values. Therefore, we evaluate different values $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ for γ and set $k = 10$. The results are shown in Table 5, where we can draw the following conclusions.

First, the results of 3D EMoSIFT and 3D MoSIFT consistently exceed traditional features (e.g., Cuboid, Harris3D and MoSIFT). More specifically, the least MLD scores (corresponding to the best recognition rate) for 3D EMoSIFT is 0.13311, compared to 0.14476 for 3D MoSIFT, 0.28064 for Cuboid, 0.18192 for Harris3D, and 0.335 for MoSIFT.

Methods \ γ	0.1	0.2	0.3	0.4	0.5
Cuboid(R)	0.36717	0.36495	0.34332	0.33111	0.31392
Cuboid(R+D)	0.33666	0.31559	0.30948	0.30782	0.28064
Harris3D hog(R)	0.30061	0.26012	0.25014	0.23516	0.23461
Harris3D hog(R+D)	0.24903	0.22795	0.22407	0.22795	0.22684
Harris3D hof(R)	0.34831	0.32668	0.31281	0.29895	0.29063
Harris3D hof(R+D)	0.32169	0.29174	0.28508	0.27898	0.27121
Harris3D hoghof(R)	0.24237	0.21963	0.20022	0.19468	0.18857
Harris3D hoghof(R+D)	0.20965	0.18802	0.18303	0.18747	0.18192
MoSIFT(R)	0.41653	0.39601	0.35885	0.36606	0.33500
MoSIFT(R+D)	0.44426	0.44260	0.43594	0.42318	0.40488
3D MoSIFT(R+D)	0.19135	0.16694	0.16195	**0.14476**	**0.14642**
3D EMoSIFT(R+D)	0.16528	**0.15419**	**0.14753**	**0.13977**	**0.13311**

Table 5: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$ and $k = 10$ ($final21 \sim final40$). It shows MLD scores by different spatio-temporal features with different values of γ, where (R) means the features are extracted from RGB video, (R+D) means the features are extracted from the RGB and depth videos. The values shown in bold indicate superior performance, with MLD scores below 0.16.

Second, from the previous works, we know that traditional features have achieved promising results (Dollár et al., 2005; Laptev, 2005; Chen and Hauptmann, 2009). However, those features may be not sufficient to capture the distinctive motion pattern only from RGB data because there is only one training sample per class.

Third, although 3D MoSIFT and 3D EMoSIFT are derived from the SIFT and MoSIFT features, MoSIFT still cannot achieve satisfactory outcomes. That is because the descriptors captured by MoSIFT are simply calculated from RGB data while 3D MoSIFT and 3D EMoSIFT construct $3D$ gradient and $3D$ motion space from the local patch around each interest point by fusing RGB-D data.

To show the distinctive views for both 3D MoSIFT and 3D EMoSIFT features, we record three gesture classes: clapping, pointing and waving. The samples are shown in Figure 15, where the training samples are shown in the first three rows (of the first two columns) and the testing samples are shown in the last three rows (of the first two columns). We first extract 3D MoSIFT and 3D EMoSIFT features from the six samples. Then we use 3D MoSIFT and 3D EMoSIFT features extracted from the three training samples to generate a codebook which has 20 visual words, respectively. Each descriptor is mapped into a certain visual word with VQ. The spatial distribution of visual words for each sample are shown in Figure 15 where different visual words are represented by different colors. It shows that 3D EMoSIFT is more compact. A more compact feature leads to a better performance (see Table 5) and can effectively reduce the redundant features (see Table 2). Besides, a compact feature should encourage the signals from the same class to have similar representations. In other words, the signals from the same class are described by similar histograms (or visual words). From the Figure 15, we can see that the samples from the same class have similar histograms (e.g., clapping gesture) when we use 3D EMoSIFT. However, 3D MoSIFT cannot

Figure 15: The first two columns are the samples used for training and testing. The third and fifth columns reveal the spatial distribution of the visual words for the samples, which show 3D EMoSIFT is more compact. We superimpose the interest points in all frames into one image. Different visual words are represented by different colors. The fourth and sixth columns are shown the histograms for each sample. The histogram vector is ℓ_2 normalization. It shows each class has some dominating visual words. A compact feature encourages gestures from the same class to be described by similar histograms (or visual words), especially the dominating visual words. The histograms from the same class learned by 3D EMoSIFT are similar (i.e., clapping gesture).

get good similar histograms. From the above discussions, we see that 3D EMoSIFT is suitable for one-shot learning gesture recognition. Interestingly, 3D EMoSIFT is also more sparsity than 3D MoSIFT (see the histograms in Figure 15)

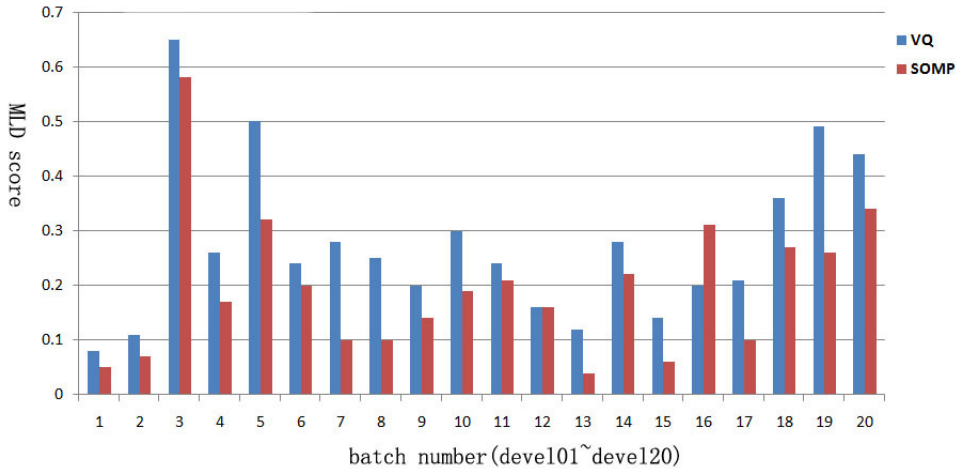

Figure 16: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$, $k = 10$ and $\gamma = 0.3$ ($devel01 \sim devel20$). The results with different coding methods (VQ, SOMP).

4.4.2 COMPARISON BETWEEN VQ AND SOMP

We then evaluate different coding methods (VQ, SOMP) on development ($devel01 \sim devel20$) batches. Figure 16 shows the results. The minimum MLD by SOMP is 0.004 (see $devel13$), while 0.008 (see $devel01$) for VQ. And most of the performances by SOMP are much better than VQ. Later, we test 3D MoSIFT and 3D EMoSIFT features on $final21 \sim final40$ batches. MLD scores are given in Table 6. It can be seen that in most cases, SOMP leads the performance whenever 3D MoSIFT or 3D EMoSIFT is used. We also provide the results by 3D EMoSIFT for every batch in Figure 17 which shows that SOMP is better than VQ in most cases. In a word, compared with VQ, SOMP not only has lower reconstruction errors (see Figure 9) but also achieves better performance. We note that 3D EMoSIFT does not work well on $devel03$ batch as shown in Figure 16. That is because there are static gestures (postures) on $devel03$ batch, while 3D EMoSIFT can only capture distinctive features when the gestures are in motion.

Methods \ γ	0.1	0.2	0.3	0.4	0.5
3D MoSIFT_VQ	0.19135	0.16694	0.16195	0.14476	0.14642
3D MoSIFT_SOMP	**0.18303**	**0.16251**	**0.15918**	0.15086	**0.14088**
3D EMoSIFT_VQ	0.16528	0.15419	0.14753	0.13977	0.13311
3D EMoSIFT_SOMP	0.17415	**0.14753**	**0.14032**	**0.13478**	**0.13145**

Table 6: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$, $k = 10$, and γ varies from 0.1 to 0.5 ($final21 \sim final40$). MLD scores are calculated by different coding methods.

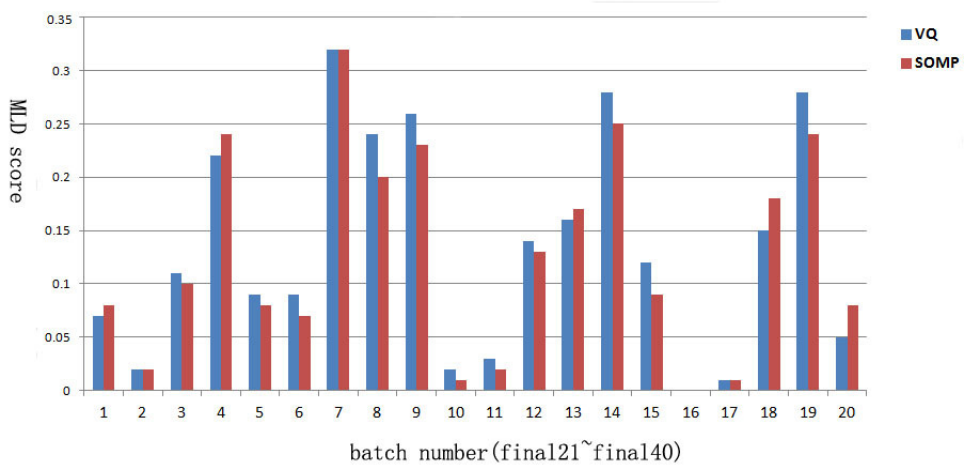

Figure 17: Parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$, $k = 10$ and $\gamma = 0.3$ ($final21 \sim final40$). The results with different coding methods (VQ, SOMP).

4.4.3 COMPARISON BETWEEN GRADIENT-BASED AND MOTION-BASED FEATURES

We know that 3D EMoSIFT feature includes two basic components, namely, gradient-based features and motion-based features. And each component is of size 384 dimensions. In this section, we separately evaluate these two components and determinate which component is more essential to gesture recognition. The results evaluated on development batches are separately shown in Figure 18 where the integrated feature consists of the gradient-based and motion-based features. The average MLD scores are 0.1945 for the integrated feature, 0.216 for the gradient-based features, and 0.313 for the motion-based features. It can be seen that the performance of the gradient-based features, which are comparative to the results of the integrated feature, are much better than the performance of the motion-based features. In addition, our method outperforms two published papers on $devel01 \sim devel20$ batches, that is say, our method: 0.1945, Lui (2012): 0.2873, Malgireddy et al. (2012): 0.2409.

As mentioned in Section 3.1, 3D EMoSIFT is constructed in two stages (interest point detection and descriptor calculation). So whenever the gradient-based or motion-based features are calculated, we should first detect the interest points. We randomly select a sample from Chalearn gesture database and test the average time with c++ programs and OpenCV library (Bradski, 2000) on a standard personal computer (CPU: 3.3GHz, RAM: 8GB). Table 7 shows that the main processing time occurs in the stage of interest point detection. The remaining parts for calculating the gradient-base and motion-based descriptor is small compared with the time for interest point detection. In our future work, we will focus on how to efficiently detect interest points.

4.4.4 COMPARISON WITH OTHER METHODS

Here, we compare the proposed approach with some popular sequence matching methods such as HMM, DTW, CRF, HCRF and LDCRF, and also give the final results of top contestants. The results are reported in Table 8 where the principal motion method (Escalante and Guyon, 2012) is the baseline method and DTW is an optional method on Chalearn gesture challenge (round 2).

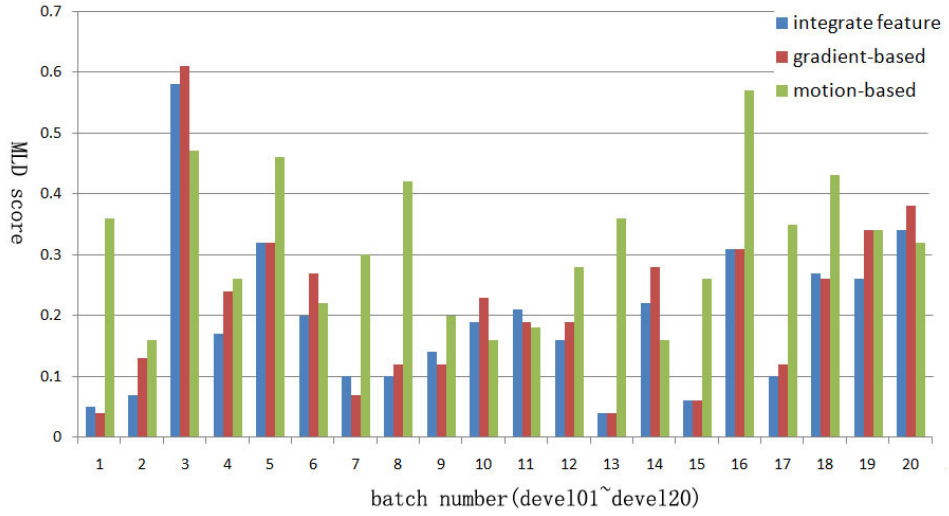

Figure 18: Results for parameters: $\beta_1 = \beta_2 = 0.005$, $\sigma = 1.6$, $s = 3$, $k = 10$, and $\gamma = 0.3$ (*devel*01 $\sim$ *devel*20).

interest point detection average time (ms/f)	gradient-based descriptor average time (ms/f)	motion-based descriptor average time (ms/f)
887	2.1	1.4

Table 7: The average computation time for different parts in 3D EMoSIFT feature.

method	validation set(01 $\sim$ 20)	final set(21 $\sim$ 40)	team name
motion signature analysis	0.0995	0.0710	Alfnie
HMM+HOGHOF	0.2084	0.1098	Turtle Tamers
BoF+3D MoSIFT	0.1824	0.1448	Joewan
principle motion	0.3418	0.3172	–
DTW	0.4616	0.3899	–
CRF	0.6365	0.528	–
HCRF	0.64	0.6	–
LDCRF	0.608	0.5145	–
our method	**0.1595**	**0.1259**	–

Table 8: Results of different methods on Chalearn gesture data set.

The top ranking results in the competition are from three teams (Alfine, Turtle Tamers and Joewan), which are provided in the technical report (Guyon et al., 2013). We use the code provided by Morency et al. (2007) to train the CRF-based classifiers, because this code was well developed and can be easily used. Every frame is represented by a vector of motion feature mentioned in Section 3.4. Those motion features extracted from training videos are used to train CRF-based

models. For the CRF model, every class has a corresponding label (gesture label). CRF predicts a label for each frame in a video. During evaluation, the video label is predicted based on the most frequently occurring label per frame (Morency et al., 2007). For the HCRF (or LDCRF) model, we train a single HCRF (or LDCRF) model with different number of hidden states (from 2 to 6 states) and select the lowest MLD scores as the final results which are shown in Table 8. We can see that the proposed method is competitive to the state-of-the-art methods. Besides, the CRF-based methods get poor performances. That is because the simple motion features may be indistinguishable to represent the gesture pattern.

5. Conclusion

In this paper, we propose a unified framework based on bag of features for one-shot learning gesture recognition. The proposed method gives superior recognition performance than many existing approaches. A new feature, named 3D EMoSIFT, fuses RGB-D data to detect interest points and constructs 3D gradient and motion space to calculate SIFT descriptors. Compared with existing features such as Cuboid (Dollár et al., 2005), Harri3D (Laptev, 2005), MoSIFT (Chen and Hauptmann, 2009) and 3D MoSIFT (Ming et al., 2012), it gets competitive performance. Additionally, 3D EMoSIFT features are scale and rotation invariant and can capture more compact and richer video representations even though there is only one training sample for each gesture class. This paper also introduces SOMP to replace VQ in the descriptor coding stage. Then each feature can be represented by some linear combination of a small number of visual codewords. Compared with VQ, SOMP leads to a much lower reconstruction error and achieves better performance.

Although the proposed method has achieved promising results, there are several avenues which can be explored. At first, most of the existing local spatio-temporal features are extracted from a static background or a simple dynamic background. In our feature research, we will focus on extending 3D EMoSIFT to extract features from complex background, especially for one-shot learning gesture recognition. Next, to speed up processing time, we can achieve fast feature extraction on a Graphics Processing Unit (GPU) (Chen et al., 2003). Also, we will explore the techniques required to optimize the parameters, such as the codebook size and sparsity.

Acknowledgments

We appreciate ChaLearn providing the gesture database (`http://chalearn.org`) whose directors are gratefully acknowledged. We would like to thank Isabelle Guyon, ChaLearn, Berkeley, California, who gives us insightful comments and suggestions to improve our manuscripts. And we are grateful to editors and anonymous reviewers whose instructive suggestions have improved the quality of this paper. Besides, thanks to acknowledge support for this project from National Natural Science Foundation (60973060, 61003114, 61172128), National 973 plans project (2012CB316304), the fundamental research funds for the central universities (2011JBM020, 2011JBM022) and the program for Innovative Research Team in University of Ministry of Education of China (IRT 201206).

References

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 994–999, 1997.

C.C. Chang and C.J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

F.S. Chen, C.M. Fu, and C.L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21:745–758, 2003.

M. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. *Technical Report*, 2009.

H. Cooper, E.J. Ong, N. Pugeault, and R. Bowden. Sign language recognition using sub-units. *Journal of Machine Learning Research*, 13:2205–2231, 2012.

A. Corradini. Dynamic time warping for off-line recognition of a small gesture vocabulary. In *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, IEEE ICCV Workshop on*, pages 82–89, 2001.

N.H. Dardas and N.D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3592–3607, 2011.

P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Proceedings of IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, 2005.

H.J. Escalante and I. Guyon. Principal motion: Pca-based reconstruction of motion histograms. *Technical Memorandum*, 2012.

L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 524–531, 2005.

F. Flórez, J.M. García, J. García, and A. Hernández. Hand gesture recognition following the dynamics of a topology-preserving network. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, pages 318–323, 2002.

P.-E. Forssen and D.G. Lowe. Shape descriptors for maximally stable extremal regions. In *Computer Vision, IEEE 11th International Conference on*, pages 1–8, 2007.

W.T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Proceedings of IEEE International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

W. Gao, G. Fang, D. Zhao, and Y. Chen. A chinese sign language recognition system based on sofm/srn/hmm. *Pattern Recognition*, 37(12):2389–2402, 2004.

T. Guha and R.K. Ward. Learning sparse representations for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1576–1588, 2012.

S. Guo, Z. Wang, and Q. Ruan. Enhancing sparsity via ℓ_p (0<p<1) minimization for robust face recognition. *Neurocomputing*, 99:592–602, 2013.

I. Guyon, V. Athitsos, P. Jangyodsuk, B. Hamner, and H.J. Escalante. Chalearn gesture challenge: Design and first results. In *Computer Vision and Pattern Recognition Workshops, IEEE Conference on*, pages 1–6, 2012.

I. Guyon, V. Athitsos, P. Jangyodsuk, H.J. Escalante, and B. Hamner. Results and analysis of the chalearn gesture challenge 2012. *Technical Report*, 2013.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of Alvey Vision Conference*, volume 15, page 50, 1988.

A. Hernández-Vela, M. A. Bautista, X. Perez-Sala, V. Ponce, X. Baró, O. Pujol, C. Angulo, and S. Escalera. Bovdw: Bag-of-visual-and-depth-words for gesture recognition. *Pattern Recognition (ICPR), 21st International Conference on*, 2012.

D. Kim, J. Song, and D. Kim. Simultaneous gesture segmentation and recognition based on forward spotting accumulative hmms. *Pattern Recognition*, 40(11):3012–3026, 2007.

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

J.F. Lichtenauer, E.A. Hendriks, and M. J T Reinders. Sign language recognition by combining statistical dtw and independent classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):2040–2046, 2008.

Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84–95, 1980.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

B.D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.

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

M.R. Malgireddy, I. Inwogu, and V. Govindaraju. A temporal bayesian model for classifying, detecting and localizing activities in video sequences. In *Computer Vision and Pattern Recognition Workshops, IEEE Conference on*, pages 43–48, 2012.

A. Malima, E. Ozgur, and M. Çetin. A fast algorithm for vision-based hand gesture recognition for robot control. In *Proceedings of IEEE Signal Processing and Communications Applications*, pages 1–4, 2006.

Y. Ming, Q. Ruan, and A.G. Hauptmann. Activity recognition from rgb-d camera with 3d local spatio-temporal features. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 344–349, 2012.

L.P. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

B.A. Olshausen, D.J. Field, et al. Sparse coding with an overcomplete basis set: A strategy employed by vi? *Vision Research*, 37(23):3311–3326, 1997.

V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:677–695, 1997.

A. Rakotomamonjy. Surveying and comparing simultaneous sparse approximation (or group-lasso) algorithms. *Signal Processing*, 91(7):1505–1526, 2011.

S. Reifinger, F. Wallhoff, M. Ablassmeier, T. Poitschke, and G. Rigoll. Static and dynamic hand-gesture recognition for augmented reality applications. In *Proceedings of the 12th International Conference on Human-computer Interaction: Intelligent Multimodal Interaction Environments*, pages 728–737, 2007.

Y. Ruiduo, S. Sarkar, and B. Loeding. Enhanced level building algorithm for the movement epenthesis problem in sign language recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

C. Shan, T. Tan, and Y. Wei. Real-time hand tracking using a mean shift embedded particle filter. *Pattern Recognition*, 40(7):1958–1970, 2007.

X. Shen, G. Hua, L. Williams, and Y. Wu. Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields. *Image and Vision Computing*, 30(3):227–235, 2012.

C. Sminchisescu, A. Kanaujia, Zhiguo Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *Computer Vision, Tenth IEEE International Conference on*, volume 2, pages 1808–1815, 2005.

H.I. Suk, B.K. Sin, and S.W. Lee. Hand gesture recognition based on dynamic bayesian network framework. *Pattern Recognition*, 43(9):3059–3072, 2010.

J. Weaver T. Starner and A. Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1371–1375, 1998.

J.A. Tropp, A.C. Gilbert, and M.J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, 1967.

C. P. Vogler. *American Sign Language Recognition: Reducing the Complexity of the Task with Phoneme-based Modeling and Parallel Hidden Markov Models*. PhD thesis, Doctoral dissertation, University of Pennsylvania, 2003.

J. Wan, Q. Ruan, G. An, and W. Li. Gesture recognition based on hidden markov model from sparse representative observations. In *Signal Processing (ICSP), IEEE 10th International Conference on*, pages 1180–1183, 2012.

H. Wang, M.M. Ullah, A. Klaser, I. Laptev, C. Schmid, et al. Evaluation of local spatio-temporal features for action recognition. In *Proceedings of British Machine Vision Conference*, 2009.

J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

S.B. Wang, A. Quattoni, L.P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1521–1527, 2006.

J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:210–227, 2009.

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

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.

M.H. Yang, N. Ahuja, and M. Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24: 1061–1074, 2002.

D. Youtian, C. Feng, X. Wenli, and Li. Yongbin. Recognizing interaction activities using dynamic bayesian network. In *Pattern Recognition, 18th International Conference on*, volume 1, pages 618–621, 2006.

Y. Zhu, G. Xu, and D.J. Kriegman. A real-time approach to the spotting, representation, and recognition of hand gestures for human–computer interaction. *Computer Vision and Image Understanding*, 85(3):189–208, 2002.