Multi-channel Shape-Flow Kernel Descriptors for Robust Video Event Detection and Retrieval

Pradeep Natarajan, Shuang Wu, Shiv Vitaladevuni, Xiaodan Zhuang, Unsang Park, Rohit Prasad, and Premkumar Natarajan

Speech, Language and Multimedia Business Unit,
Raytheon BBN Technologies, Cambridge, MA 02138
{pradeepn,swu,svitalad,xzhuang,upark,rprasad,pnatarajan}@bbn.com

Abstract. Despite the success of spatio-temporal visual features, they are hand-designed and aggregate image or flow gradients using a pre-specified, uniform set of orientation bins. Kernel descriptors [1] generalize such orientation histograms by defining match kernels over image patches, and have shown superior performance for visual object and scene recognition. In our work, we make two contributions: first, we extend kernel descriptors to the spatio-temporal domain to model salient flow, gradient and texture patterns in video. Further, we apply our kernel descriptors to extract features from different color channels. Second, we present a fast algorithm for kernel descriptor computation of $O(1)$ complexity for each pixel in each video patch, producing two orders of magnitude speedup over conventional kernel descriptors and other popular motion features. Our evaluation results on TRECVID MED 2011 dataset indicate that the proposed multi-channel shape-flow kernel descriptors outperform several other features including SIFT, SURF, STIP and Color SIFT.

Keywords: Kernel Descriptor, Multi-channel, Low Level Feature, Video Event Detection, TRECVID.

1 Introduction

The widespread availability of cheap hand-held cameras and video sharing websites such as YouTube has resulted in massive amounts of video content online. The ability to rapidly analyze and summarize content from such videos entails a wide range of applications. Significant effort has been made in recent literature to develop such techniques [2][3][4][5]. However, the sheer volume of such content as well as the challenges in analyzing videos introduce significant scalability challenges in applying successful bag-of-words approaches [6] used in image retrieval.

Features such as STIP [7] and HoG3D [8] that extend image level features to the spatio-temporal domain have shown promise in recognizing actions from unstructured videos [8][9]. These features discretize the gradient or optical flow

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part II, LNCS 7573, pp. 301–314, 2012.

orientations into a d-dimensional indicator vector $\delta(z)=[\delta_1(z),\ldots,\delta_d(z)]$ with

$$\delta_i(z) = \begin{cases} 1 & \text{if } \lfloor \frac{d\theta(z)}{2\pi} \rfloor = i-1 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Despite their success, these features are hand designed and do not utilize full information available in measuring patch similarity. In recent work, several efforts have been made to develop principled approaches to design and learn such low-level features. In [10], a convolutional GRBM method was proposed to extract spatio-temporal features using a multi-stage architecture. In [11], a convolutional independent subspace analysis (ISA) network was proposed to extract patch level features from pixel attributes.

These deep learning approaches are in effect mapping pixel attributes into patch level features using a hierarchical architecture. In [12], a two layer hierarchical sparse coding scheme is used for learning image representations at the pixel level. The orientation histogram in (1) in effect uses a pre-defined d-dimensional codebook that divides the θ space into uniform bins, and uses hard quantization for projecting pixel gradients. In contrast [12] allows data driven learning of pixel level dictionaries, and the pixel features are projected to the learnt dictionary using sparse coding to get a vector $W(z)=(w_1(z),\ldots,w_d(z))$. After pooling such pixel level projections within local regions, the first layer codes are passed to the second layer for jointly encoding signals in the region. The orientation histograms and hierarchical sparse coding in effect define the following kernel for measuring the similarity between two patches P and Q:

$$K(P,Q) = F_h(P)^\top F_h(Q) = \sum_{z\in P}\sum_{z'\in Q} \tilde{m}(z)\tilde{m}(z')\Phi(z)^\top\Phi(z') \quad (2)$$

where

- $F_h(P)=\sum_{z\in P}\tilde{m}(z)\Phi(z)$ is the patch sum
- $\tilde{m}(z)=m(z)/\sqrt{\sum_{z\in P} m(z)^2+\epsilon_g}$ is the normalized gradient magnitude with ϵ_g a small constant, and
- $\Phi(z)=\delta(z)$ for HoG and $\Phi(z)=W(z)$ for hierarchical sparse coding.

The kernel descriptors proposed in [1] generalize these approaches by replacing the product $\Phi(z)^\top\Phi(z')$ in equation (2) with a match kernel $k(z,z')$ and allows us to induce arbitrary feature spaces $\Phi(z)$ (including infinite dimensional) from pixel level attributes. This provides a powerful framework for designing rich low-level features and has shown state-of-the-art results for image and object recognition [1][13].

A crucial limitation of kernel descriptors is that kernel computations are costly and hence it is slow to extract them from densely sampled video patches. In our work, we present a fast algorithm for kernel descriptor computation that takes $O(1)$ operations per pixel in each patch, based on pre-computed kernel values. This speeds up the kernel descriptor features under consideration, to levels that are comparable with D-SIFT[14] and color SIFT[15], and two orders of magnitude faster than STIP[16] and HoG3D[8]. In contrast, the kernel descriptor

computation in [1] is 4× and 10×slower than SIFT, for gradient and LBP based kernel descriptors respectively. Furthermore, we apply kernel descriptors to extract gradient, flow and texture based features for video analysis. We test our approach on a large database of internet videos used in the TRECVID MED 2011 evaluations, and compare the proposed kernel descriptors with a large set of image and motion based features. Our flow based kernel descriptors are up to two orders of magnitude faster than STIP and HoG3D, and also produce significant performance improvements. Further, using features from multiple color planes produces small but consistent gains.

The rest of the paper is organized as follows - we provide an overview of kernel descriptors in section 2, describe video specific features we extract in our experiments in section 3, present a fast feature computation algorithm in section 4, describe feature representation and early and late fusion techniques we use in section 5, and present experimental results in section 6.

2 Kernel Descriptors

Kernel descriptors [1] provide a unified framework for turning pixel level attributes such as gradients into patch level features, by highlighting the kernel view of orientation histogram features such as SIFT and HoG. Feature extraction using kernel descriptors involves three steps: (1) designing kernels for matching patches using pixel attributes; (2) learning a compact set of basis vectors using kernel principal component analysis (KPCA); (3) constructing kernel descriptors by projecting the infinite-dimensional feature vectors induced by pixel attributes to the learned basis vectors.

Following the notations in [1], the gradient match kernel $K_{grad}(P, Q)$ measures the similarity between patches P and Q based on the pixel gradient attribute:

$$K_{grad}(P,Q) = \sum_{z \in P} \sum_{z' \in Q} \tilde{m}(z)\tilde{m}(z')k_o(\tilde{\theta}_z, \tilde{\theta}_{z'})k_p(z,z') \tag{3}$$

where z denotes the 2D position of a pixel in an image patch normalized to [0,1] and $\tilde{\theta}_z$ is the normalized gradient vector defined as:

$$\tilde{\theta}_z = [sin(\theta(z)), cos(\theta(z))] \tag{4}$$

The position kernel k_p and orientation kernel k_o are defined using Gaussian kernels as:

$$\begin{aligned} k_p(z,z') &= e^{-\gamma_p||z-z'||^2} \\ k_o(\tilde{\theta}_z, \tilde{\theta}_{z'}) &= e^{-\gamma_o||\tilde{\theta}(z)-\tilde{\theta}(z')||^2} \end{aligned} \tag{5}$$

Based on the patch similarity kernel defined in equation (3), kernel descriptors extract compact low-dimensional features by sampling sufficient basis vectors

uniformly and densely from each pixel attribute's support region, and then learning compact basis vectors using KPCA. Thus, the gradient kernel descriptor for a patch P will have the form:

$$F^t_{grad}(P) = \sum_{i=1}^{d_o} \sum_{j=1}^{d_p} \alpha^t_{ij} \left\{ \sum_{z \in P} \tilde{m}(z) k_o(\tilde{\theta}(z), x_i) k_p(z, y_j) \right\} \tag{6}$$

where $\{x_i\}_{i=1}^{d_o}$ and $\{y_j\}_{j=1}^{d_p}$ are uniformly sampled from the corresponding support regions and d_o and d_p are the sizes of the basis vectors for the orientation and position kernels respectively. Similarly, a match kernel based on local binary pattern (LBP)[17] is defined in [1] as:

$$F^t_{LBP}(P) = \sum_{i=1}^{d_l} \sum_{j=1}^{d_p} \alpha^t_{ij} \left\{ \sum_{z \in P} \tilde{s}(z) k_l(b(z), x_i) k_p(z, y_j) \right\} \tag{7}$$

where $\tilde{s}(z)=s(z)/\sqrt{\sum_{z \in P} s(z)^2 + \epsilon_s}$ is the standard deviation of the pixel values in the 3×3 neighborhood of z, ϵ_s is a small constant and $b(z)$ is the 8-dimensional LBP vector at z as defined in [17]. The set of all possible 2^8=256 LBP vectors is chosen as the basis vector set and k_l is defined using an RBF kernel similar to 5.

The coefficients α^t_{ij} are learned through kernel principal component analysis as follows:

$$\begin{aligned} \mathbf{K}_{ijst} &= k_o(x_i, x_j) k_p(y_s, y_t) \\ \mathbf{K}_c &= \mathbf{K} - \mathbf{1}_N \mathbf{K} - \mathbf{K} \mathbf{1}_N + \mathbf{1}_N \mathbf{K} \mathbf{1}_N \\ \mathbf{K}_c \alpha^t &= \lambda^t \alpha^t \end{aligned} \tag{8}$$

where $\mathbf{K}$ is the kernel matrix, $\mathbf{K}_c$ is the centered kernel matrix, and 1_N denotes a $N \times N$ matrix with each element taking value $1/N$. Thus, α^t correspond to the eigenvectors of the centered kernel matrix $\mathbf{K}_c$.

3 Multi-channel Gradient Flow Kernel Descriptors

We extract kernel descriptors using image and optical flow gradients from a dense grid of spatio-temporal patches. In our experiments we consider the following six features.

Grayscale Gradient Descriptors (Gray KDES-G): These features are extracted from the gradients L_x and L_y computed along the x and y directions, on gray scale image frames sampled from a video. We extract kernel descriptors based on equation (6) for each patch.

Grayscale Flow Descriptors (Gray KDES-F): These features are extracted from the optical flow fields F_x and F_y computed from adjacent frames in different temporal locations of the video. F_x and F_y are then used to estimate orientations θ at different pixel locations for computing the gradient kernel descriptors.

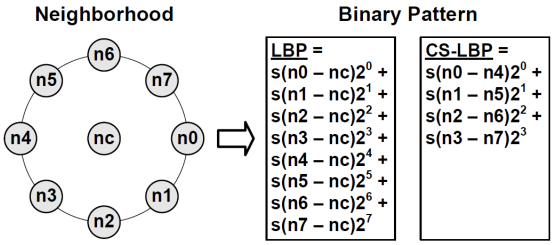

Fig. 1. Schematic comparing LBP and CS-LBP [18]

Grayscale Gradient+Flow Descriptors (Gray KDES-FG): These features combine gradient and flow information, by concatenating the Gray KDES-G and Gray KDES-F descriptors at each video patch.
Grayscale LBP Descriptors (Gray KDES-L): We extracted LBP features [17] from the intensity values of grayscale frames and used them within the kernel descriptor framework as described in [1].
Grayscale Center Surround LBP Descriptors (Gray KDES-CL): The number of basis vectors for the joint LBP-position kernel in equation (7) is 256×25=6400. Applying KPCA on the joint set results in too many components that slows down feature extraction, while using a fixed, small number of components (such as 200 in [1]) results in poor approximation of the space. In our work we used a variant of LBP called center-symmetric LBP (CS-LBP)[18] that extracts a 4-dimensional binary vector from a pixel's neighborhood (See Figure 1.). This space has 16×25=400 dimensions and KPCA produces better approximation of the space with fewer principal components.
Color Gradient Descriptors (Color KDES-G): We first split the frame images sampled from video to constituent color planes. In our experiments we used the (R,G,B) planes, but we can use other color channels too. We then extract KDES-G features from the gradients computed on each plane. For each patch, we concatenate the KDES-G features from the different color planes.
Color Flow Descriptors (Color KDES-F): We compute these features from the optical flow fields computed in each color plane and then concatenate the KDES-F features from different color planes for each video patch.
Color Gradient+Flow Descriptors (Color KDES-FG): For these features we concatenate the Color KDES-G and Color KDES-F features for each video patch.
Color LBP Descriptors (Color KDES-L): These features extract KDES-L features from each of the (R,G,B) channels.
Color Center Surround LBP Descriptors (Color KDES-CL): These features are similar to Gray KDES-CL, but are extracted from each of the (R,G,B) channels.
For our experiments, we uniformly sample every 50 frames from each video, and compute the descriptors from patches of size 16×16×1, 25×25×1 and 31×31×1 on an evenly spaced grid of 8 pixels.

4 Fast Computation of Kernel Descriptors

The major bottleneck in computing kernel descriptors is in evaluating the kernel function $k_o k_p$ in equation (6). The naive approach of computing the product for each (x_i,y_j) would cost $d_o d_p$ kernel computations. We can minimize the number of calls to costly kernel computation functions by computing the two kernel values separately at cost d_o+d_p, but computation of the sum in equation (6) still costs $O(d_o d_p)$ for each pixel $z \in P$. In this section we will describe our approach to compute the kernel sum at each pixel in $O(1)$ time.

Let $\mathbf{K}_o$ and $\mathbf{K}_p$ be the kernel matrices defined over the orientation and position basis vectors respectively:

$$\mathbf{K}_{o,ij} = k_o(x_i, x_j) \quad \mathbf{K}_{p,st} = k_p(y_s, y_t) \tag{9}$$

Let $\mathbf{K}_{o,c}$ and $\mathbf{K}_{p,c}$ denote the corresponding centered orientation and position kernel matrices. Then from the definitions in (8) and (9) we have

$$\mathbf{K}_{o,c} \otimes \mathbf{K}_{p,c} \alpha^t = \lambda^t \alpha^t \tag{10}$$

where $\otimes$ denotes the Kronecker product. Since the kernel matrices $\mathbf{K}_{o,c}$ and $\mathbf{K}_{p,c}$ are symmetric positive definite, we have

$$\begin{aligned} \mathbf{K}_{o,c} \otimes \mathbf{K}_{p,c} &= [U_{o,c}^\top S_{o,c} U_{o,c}] \otimes [U_{p,c}^\top S_{p,c} U_{p,c}] \\ &= [U_{o,c} \otimes U_{p,c}]^\top [S_{o,c} \otimes S_{p,c}][U_{o,c} \otimes U_{p,c}] \end{aligned} \tag{11}$$

This indicates that the eigenvectors of $\mathbf{K}_{o,c} \otimes \mathbf{K}_{p,c}$ can be computed from the Kronecker product of the eigenvectors of $\mathbf{K}_{o,c}$ and $\mathbf{K}_{p,c}$ and the eigenvalues of $\mathbf{K}_{o,c} \otimes \mathbf{K}_{p,c}$ are the product of the eigenvalues of $\mathbf{K}_{o,c}$ and $\mathbf{K}_{p,c}$.

This factorization was used in [13] for fast computation of the eigenvectors of the Kronecker product of kernel matrices. We utilize the same factorization in the sum in equation (6) for fast computation of the kernel descriptors. Here each eigenvector α^t in equation (8) can be written as the Kronecker product of corresponding orientation and position eigenvectors $\alpha^t = \alpha_o^t \otimes \alpha_p^t$, and each coefficient α_{ij}^t in (6) can be represented by the product $\alpha_{o,i}^t \alpha_{p,j}^t$. Substituting in (6) we have:

$$F_{grad}^t(P) = \sum_{i=1}^{d_o} \sum_{j=1}^{d_p} \alpha_{o,i}^t \alpha_{p,j}^t \left\{ \sum_{z \in P} \tilde{m}(z) k_o(\tilde{\theta}(z), x_i) k_p(z, y_j) \right\} \tag{12}$$

Rearranging, we get

$$F_{grad}^t(P) = \sum_{z \in P} \tilde{m}(z) \left\{ \sum_{i=1}^{d_o} \alpha_{o,i}^t k_o(\tilde{\theta}(z), x_i) \right\} \left\{ \sum_{j=1}^{d_p} \alpha_{p,j}^t k_p(z, y_j) \right\} \tag{13}$$

With this factorization, we can compute the inner sums over x_i and y_j separately at cost d_o+d_p per pixel, in each patch P. We can achieve further speed

up by pre-computing the inner sums for a large number of orientation and position values and storing them in look-up tables T_o^t and T_p^t respectively, for each KPCA component t. Given an orientation $\theta(z)$ and position z, we can simply retrieve from T_o^t and T_p^t, the values corresponding to the nearest pre-computed orientation and position:

$$F_{grad}^t(P) = \sum_{z \in P} \tilde{m}(z) T_o^t(\theta(z)) T_p^t(z) \tag{14}$$

This takes $O(1)$ for each pixel z in patch P. In our experiments, we use 25 basis vectors x_i for the orientation, and sample the position basis vectors using a 5×5 grid. Thus the fast kernel descriptor computation in (14) produces a 625× speed-up over the naive computation in (6) and 50× speedup over the factorized sum computation in (13). We pre-compute orientation sums for n_o=1000 $\theta(z)$ values in the range $[0, 2\pi]$, n_l=16 possible values for CS-LBP vectors and n_p=1000 points for $z \in [0,1] \times [0,1]$.

5 Feature Representation and Fusion

We use the popular bag-of-words framework to represent the information from different feature descriptors. This is done in two steps - in the first *coding* step the descriptors are projected to a pre-trained codebook of descriptor vectors, and then in the *pooling* step the projections are aggregated to a fixed length feature vector. We use both spatial [19] and spatio-temporal [7] pooling. From these features, we further employ kernel based fusion and score level fusion to achieve more robust performance.

5.1 Coding and Pooling Strategies

Formally, we present a video by a set of low-level descriptors, x_i, where $i \in \{1..\mathrm{N}\}$ is the set of locations. Let M denote the different spatial/spatio-temporal regions of interest, and N_m denote the number of descriptors extracted within region m. Let f and g denote the coding and pooling operators respectively. Then, the vector z representing the entire video is obtained by sequentially coding and pooling over all regions and concatenating:

$$\alpha_i = f(x_i), \quad i = 1..N \tag{15}$$

$$\mathbf{h}_m = g(\{\alpha_i\}_{i \in N_m}), \quad m = 1, ..., M \tag{16}$$

$$\mathbf{z}^T = [h_1^T ... h_M^T] \tag{17}$$

Coding: For the coding step, we first learn a codebook using k-means or a similar unsupervised clustering algorithm from a sample set of feature vectors. In hard quantization, we assign each feature vector x_i to the nearest codeword from the codebook as

$$\alpha_i \in \{0,1\}^K, \quad \alpha_{i,j} = 1 \Leftrightarrow j = \arg\min_{k \leq K} ||x_i - c_k||^2 \tag{18}$$

where c_k is the k^{th} codeword. In soft quantization [20], the assignment of the feature vectors to codewords is distributed as

$$\alpha_{i,j} = \frac{\exp\left(-\beta||x_i - c_j||^2\right)}{\sum_{k=1}^{K} \exp\left(-\beta||x_i - c_k||^2\right)} \tag{19}$$

where β controls the soft assignment. In our experiments we find soft quantization to consistently outperform hard quantization.

Pooling: The two most popular pooling strategies are average and max. In average pooling, we take the average of the α_i assigned to different codewords for different feature vectors as $\mathbf{h} = 1/N \sum_{i=1}^{N} \alpha_i$. In max pooling, we take the maximum of the α_i's as $\mathbf{h} = \max_{i=1..N} \alpha_i$. In our work, we find average pooling to consistently outperform max pooling for video retrieval. Further spatial pooling with 1×1+2×2+1×3 partition of the (x, y) space has consistently superior performance for all the features considered. This is similar to previous results that show improvements with spatial [19] and spatio-temporal [7] feature pooling.

5.2 Kernel-Based Feature Fusion

We combine multiple features in an early fusion framework by using p-norm Multiple Kernel Learning (MKL)[21], with p>1. For each feature, we first compute exponential χ^2 kernels, defined by $K(\mathbf{x}, \mathbf{y}) = e^{-\rho \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}}$ for each pair of samples $\mathbf{x}$ and $\mathbf{y}$ in the training set. Then, given a set of kernels $\{K_k\}$ for individual features, we learn a linear combination $K = \sum_k d_k K_k$ of the base kernels. The primal of this problem can be formulated as

$$\begin{aligned} \min_{w,b,\xi \geq 0, d \geq 0} \frac{1}{2} \sum_k \mathbf{w}_k^t \mathbf{w}_k + C \sum \xi_i + \frac{\lambda}{2} (\sum_k d_k^p)^{\frac{2}{p}} \\ \text{s.t. } y_i (\sum_k \sqrt{d_k} \mathbf{w}_k^t \phi_k(x_i) + b) \geq 1 - \xi_i \end{aligned} \tag{20}$$

The convex form of (20) is obtained by substituting w_k for $\sqrt{d_k} w_k$. To solve (20) efficiently, we use Sequential Minimal Optimization (SMO) [22]. This is possible by first computing the Lagrangian

$$\begin{aligned} L = \frac{1}{2} \sum_k \mathbf{w}_k^t \mathbf{w}_k / d_k + \sum (C - \beta_i) \xi_i + \frac{\lambda}{2} (\sum_k d_k^p)^{\frac{2}{p}} \\ - \sum_i \alpha_i [y_i (\sum_k \mathbf{w}_k^t \phi_k(x_i) + b) - 1 + \xi_i] \end{aligned} \tag{21}$$

and then computing the l_p-MKL dual as

$$D = \max_{\alpha \in A} \mathbf{1}^t \alpha - \frac{1}{8\lambda} (\sum_k (\alpha^t H_k \alpha)^q)^{\frac{2}{q}} \tag{22}$$

where $\frac{1}{p} + \frac{1}{q} = 1$, $A = \{\alpha \mid 0 \leq \alpha \leq C\mathbf{1}, \mathbf{1}^t Y\alpha = 0\}$, $H_k = YK_kY$, and Y is a diagonal matrix with labels on the diagonal. The kernel weights can then be computed as

$$d_k = \frac{1}{2\lambda}\left(\sum_k (\alpha^t H_k \alpha)^q\right)^{\frac{1}{q}-\frac{1}{p}} (\alpha^t H_k \alpha)^{\frac{q}{p}} \tag{23}$$

Since the dual objective (22) is differentiable with respect to α, the SMO algorithm can be applied by selecting two variables at a time and optimizing until convergence.

5.3 Score Level Late Fusion

We adopted a weighted average fusion strategy that assigns video specific weights based on each system's detection threshold. This is based on the intuition that a system has low confidence when its score for a particular video is close to the detection threshold, and high confidence when the scores are significantly different from the threshold. Given the confidence score p_i from system i for a particular video, the weight for that system and video is computed as:

$$w_i = \begin{cases} \frac{Th_i - p_i}{Th_i} & \text{if } p_i < Th_i \\ \frac{p_i - Th_i}{1 - Th_i} & \text{else} \end{cases} \tag{24}$$

where Th_i is the detection threshold. The final score P for a video is computed as $P = \sum_i w_i p_i / \sum_i w_i$. In our experiments, this approach consistently improved performance over any individual system.

6 Experimental Results

For our experiments, we used the dataset released as part of the TRECVID MED 11 evaluations (http://www.nist.gov/itl/iad/mig/med11.cfm). This set includes event kits containing ≈100-200 videos for each of the 10 test events of interest, and the DEV-T set containing 10723 background videos. The 10 events of interest include the following - *birthday party (BP)*, *changing a vehicle tire (CT)*, *flash mob gathering (FM)*, *getting a vehicle unstuck (GV)*, *grooming an animal (GA)*, *making a sandwich (MS)*, *parade (PA)*, *parkour (PK)*, *repairing an appliance (RA)*, *working on a sewing project (WS)*. Figure 2 shows sample frames from each of the events of interest.

A key challenge in web video retrieval is the large imbalance in the available training data for positive and negative examples. In our experiments on the TRECVID MED11 dataset, the ratio of positive to negative examples is about 1:49. Since the MKL optimization function in equation (20) optimizes for accuracy, the solution often converges to the trivial classifier that declares all examples as negative, which has 98% accuracy in our case. To address this, we train models for combining multiple features by performing an extensive grid

search over C and p. At each parameter setting, we perform a k-fold validation on the available training data and estimate a threshold that minimizes the Normalized Detection Cost (NDC) score that is defined as:

$$f = \min_{Th} \{w_{MD} P_{MD}(Th) + w_{FA} P_{FA}(Th)\}, \quad P_{MD}(Th) < 0.75 \tag{25}$$

where Th is the detection threshold, $P_{MD}(Th)$ and $P_{FA}(Th)$ are the missed detection and false alarm rates at the detection threshold, and w_{MD}, w_{FA} are the relative weights for missed detections and false alarms. For the TRECVID MED dataset, different systems are compared with $w_{MD} = 1.0$, $w_{FA} = 12.49$. A similar training regime can also be used to minimize other metrics such as the area under curve (AUC), and mean average precision (MAP).

We compared the performance of KDES (Kernel Descriptor) features with a large set of image, color and flow based features. For image gradient based features, we compared with SURF[23], SIFT[24], D-SIFT[14], and CHoG[25]. For color features, we considered RGB-SIFT and C-SIFT[15]. For motion based features, we compared with STIP[7], HoGHoF3D[8] and ISA[11]. Table 1 compares the performance of the KDES features with different state-of-the-art features, for each of 10 events of interest. The proposed KDES features are the single best features for 7 of the 10 classes.

The Gray KDES-FG features that extract pixel level flow and gradient information similar to STIP and HoGHoF3D significantly outperform these features, and are also 2 orders of magnitude faster. Further, KDES-FG also outperforms the ISA features, which learn spatio-temporal patch features from pixel attributes using deep learning. The color KDES-G features outperform the RGB-SIFT and C-SIFT features that also extract gradient patterns from different color planes. The grayscale KDES-G features perform similarly to D-SIFT at approximately the same speed. The KDES-L features have weaker performance, primarily due to the small number of principal components chosen during KPCA out of the 6400 dimensional basis vector space. However, the KDES-CL features based on the more compact CS-LBP features produce comparable performance to KDES-G features. We see improved performance with KDES features over a range of missed detection and false alarm rate operating points on the DET

Fig. 2. Example video frames from each of the 10 test events

Table 1. Comparative Performance of KDES Features (using NDC score), and speed (fps) measured on 640×360 pixel resolution video, for each patch size at which features are extracted

	BP	CT	FM	GV	GA	MS	PA	PK	RA	WS	Mean	Speed
SURF[23]	0.98	**0.77**	0.44	0.85	1.04	1.34	0.87	0.82	0.81	0.79	0.91	0.600
SIFT[24]	1.07	0.90	0.51	0.95	0.91	1.57	0.89	0.79	0.73	0.93	0.93	0.530
D-SIFT[14]	0.84	0.82	0.45	0.82	0.98	1.30	0.91	0.78	0.73	0.67	0.83	0.250
CHoG[25]	1.01	0.92	0.43	0.91	1.39	1.57	0.91	0.87	0.76	0.97	0.97	0.177
RGB-SIFT[15]	0.80	0.81	0.40	0.85	1.09	1.51	0.91	0.75	0.75	0.80	0.87	0.019
C-SIFT[15]	0.90	1.02	0.44	0.88	1.16	1.11	0.92	0.79	0.67	0.83	0.87	0.019
STIP[7]	1.16	1.14	0.48	1.06	1.18	1.27	**0.84**	0.77	0.64	**0.54**	0.91	0.008
HoGHoF3D[8]	1.09	1.14	0.48	1.30	1.14	1.43	0.92	0.76	0.58	0.76	0.96	0.002
ISA[11]	1.42	1.04	0.64	0.99	1.18	1.40	0.99	1.02	0.64	1.17	1.05	0.001
Gray KDES-F	1.11	0.94	0.45	1.05	1.15	1.41	0.86	0.89	0.66	0.71	0.92	0.230
Gray KDES-G	0.90	0.90	0.41	0.78	0.90	1.25	0.92	0.76	0.67	0.76	0.83	0.240
Gray KDES-FG	0.88	0.88	0.39	0.83	0.96	**1.07**	0.93	**0.72**	0.62	0.64	**0.79**	0.120
Gray KDES-L	0.88	1.02	0.47	1.08	1.35	1.39	0.93	0.84	0.80	0.84	0.96	0.180
Gray KDES-CL	0.90	0.89	0.38	0.85	0.91	1.36	0.89	0.80	0.71	0.68	0.84	0.180
Color KDES-F	1.20	0.99	0.45	1.15	0.96	1.22	0.95	0.83	**0.54**	0.84	0.91	0.072
Color KDES-G	0.80	0.85	0.41	**0.73**	**0.83**	1.41	0.93	0.76	0.69	0.66	0.81	0.084
Color KDES-FG	**0.78**	0.86	0.40	0.80	0.95	1.2	0.87	0.73	0.66	0.66	**0.79**	0.036
Color KDES-L	0.96	0.97	0.44	0.98	1.21	1.23	1.05	0.88	0.71	0.73	0.92	0.060
Color KDES-CL	0.89	0.89	**0.36**	0.89	0.96	1.43	0.88	0.83	0.67	0.62	0.84	0.060

curve. This is illustrated in figures 3 and 4 for two of the events considered. The processing speed is also comparable with conventional features (Shown in the last column of Table 1).

Fusion Experiments: We next experimented with different combinations of KDES features. We combined the KDES-F and KDES-G that extract pixel level information from flow and gradient patterns, and with KDES-FG that extracts pixel level information jointly from flow and gradients. We also tested the performance of including KDES-CL features for fusion. We considered feature fusion using both early fusion (MKL) and score level fusion. Table 2 summarizes the results of our fusion experiments. KDES-F+KDES-G fusion produces comparable performance to KDES-FG. Combining all the features produces better performance for both grayscale and color features. In addition, combining grayscale

Table 2. Comparison of different fusion strategies of KDES features (using NDC score)

	Individual KDES Features				MKL			Score Fusion		
	F	G	FG	CL	F+G	F+G+ FG	F+G+ FG+CL	F+G	F+G+ FG	F+G+ FG+CL
Gray	0.9238	0.8262	0.7927	0.8362	0.8238	0.7852	0.7871	0.7758	0.7554	0.7623
Color	0.9123	0.8077	0.7908	0.8413	0.7809	0.7860	0.7863	0.7822	0.7490	0.7459
Combined						0.7787	0.7904		0.7379	**0.7374**

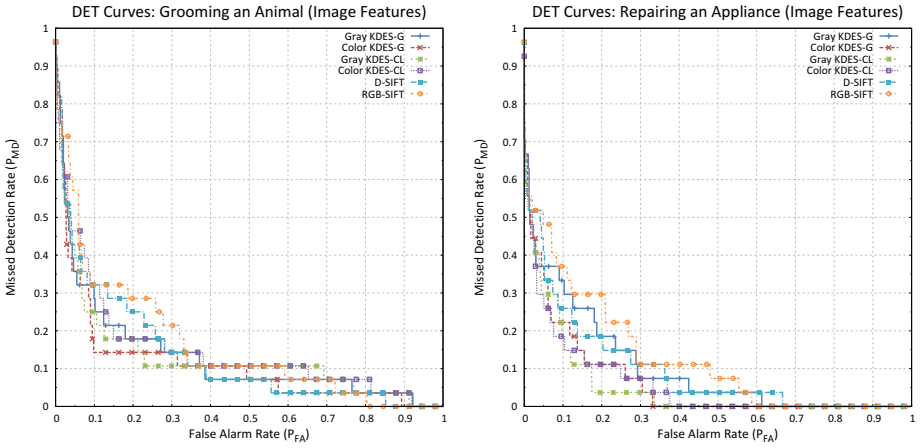

Fig. 3. DET curves comparing image features for classes "Grooming an animal" and "Repair an appliance"

Table 3. Fusion of KDES features with standard features

Feature Combination	NDC
D-SIFT+RGB-SIFT+STIP+HoGHoF3D	0.7539
All Individual KDES	0.7374
All Individual KDES+D-SIFT+RGB-SIFT+ STIP+HoGHoF3D	**0.7148**

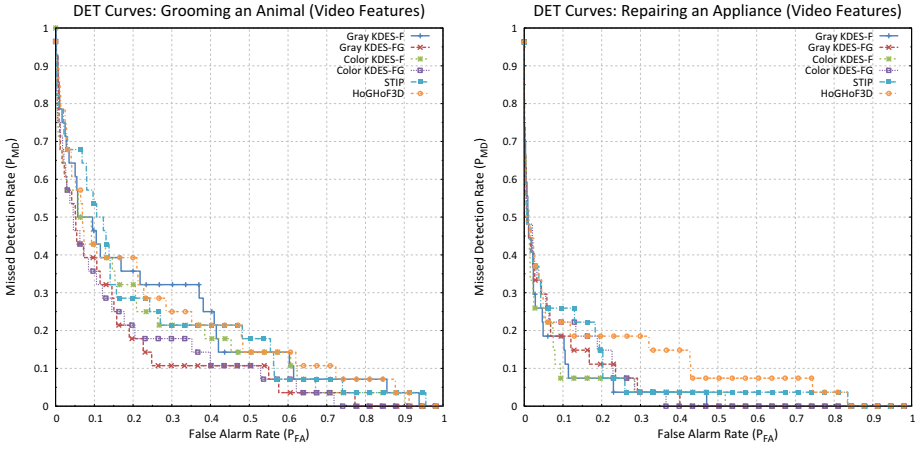

Fig. 4. DET curves comparing motion features for classes "Grooming an animal" and "Repair an appliance"

and color features produces further gains indicating presence of complementary information in the two types of features. Further, score level fusion generally yields better performance than MKL for the different combinations considered.

Next, we combined our KDES features with a set of standard features. We chose D-SIFT, RGB-SIFT, STIP and HoGHoF3D since they extract information similar to Gray KDES-G, Color KDES-G and Gray KDES-FG features. Table 3 compares the performance of different combinations. The KDES feature combination outperforms the D-SIFT+RGB-SIFT+STIP+HoGHoF3D system, while combining all the KDES and standard features produces additional gains, indicating presence of complementary information from the KDES features on top of the standard feature set.

7 Conclusion

We present a set of features that extend kernel descriptors for analyzing videos, and present an efficient algorithm for rapid computation of such features in videos. Together, this allows us to define a rich set of pixel level spatio-temporal features for robust video analysis. We rigorously analyzed their performance on large benchmark video event detection dataset used in the TRECVID MED 2011 evaluations. Our results indicate that kernel descriptors significantly outperform popular motion features such as STIP, HoGHoF3D and the recently developed ISA features. Furthermore, incorporating color information produces small performance improvements at the cost of increased computation. In the future, within the kernel descriptor framework, we aim to explore a larger set of pixel attributes as well as develop hierarchical architectures for learning video level features.

Acknowledgment. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via the Department of Interior National Business Center contract number D11PC20071. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC or the U.S. Government.

References

1. Bo, L., Ren, X., Fox, D.: Kernel descriptors for visual recognition. In: NIPS, pp. 244–252 (2010)
2. Zanetti, S., Zelnik Manor, L., Perona, P.: A walk through the web's video clips. In: InterNet, pp. 1–8 (2008)
3. Song, Y., Zhao, M., Yagnik, J., Wu, X.: Taxonomic classification for web-based videos. In: CVPR, pp. 871–878 (2010)
4. Wang, Z., Zhao, M., Song, Y., Kumar, S., Li, B.: Youtubecat: Learning to categorize wild web videos. In: CVPR, pp. 879–886 (2010)

5. Toderici, G., Aradhye, H., Pasca, M., Sbaiz, L., Yagnik, J.: Finding meaning on youtube: Tag recommendation and category discovery. In: CVPR, pp. 3447–3454 (2010)
6. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, pp. 1–22 (2004)
7. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR, pp. 1–8 (2008)
8. Wang, H., Ullah, M.M., Kläser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC (2009)
9. Liu, J., Luo, J., Shah, M.: Recognizing realistic actions from videos in the wild. In: CVPR (2009)
10. Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C.: Convolutional Learning of Spatio-temporal Features. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 140–153. Springer, Heidelberg (2010)
11. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: CVPR, pp. 3361–3368 (2011)
12. Yu, K., Lin, Y., Lafferty, J.D.: Learning image representations from the pixel level via hierarchical sparse coding. In: CVPR, pp. 1713–1720 (2011)
13. Bo, L., Lai, K., Ren, X., Fox, D.: Object recognition with hierarchical kernel descriptors. In: CVPR, pp. 1729–1736 (2011)
14. Boureau, Y., Bach, F., Le Cun, Y., Ponce, J.: Learning mid-level features for recognition. In: CVPR, pp. 2559–2566 (2010)
15. van de Sande, K.E.A., Gevers, T., Snoek, C.G.M.: Evaluating color descriptors for object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 1582–1596 (2010)
16. Laptev, I.: On space-time interest points. International Journal of Computer Vision 64, 107–123 (2005)
17. Ojala, T., Pietikainen, M., Maenpa, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. PAMI 24, 971–987 (2002)
18. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of Interest Regions with Center-Symmetric Local Binary Patterns. In: Kalra, P.K., Peleg, S. (eds.) ICVGIP 2006. LNCS, vol. 4338, pp. 58–69. Springer, Heidelberg (2006)
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, vol. II, pp. 2169–2178 (2006)
20. van Gemert, J.C., Veenman, C.J., Smeulders, A.W.M., Geusebroek, J.M.: Visfual word ambiguity. PAMI 32(7), 1271–1283 (2010)
21. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: lp-norm multiple kernel learning. Journal of Machine Learning Research 12, 953–997 (2011)
22. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization, pp. 185–208. MIT Press (1999)
23. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. CVIU 110, 346–359 (2008)
24. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV 60, 91–110 (2004)
25. Chandrasekhar, V., Takacs, G., Chen, D.M., Tsai, S.S., Grzeszczuk, R., Girod, B.: Chog: Compressed histogram of gradients a low bit-rate feature descriptor. In: CVPR, pp. 2504–2511 (2009)