NANYANG TECHNOLOGICAL UNIVERSITY

Nanyang Technological University

Final year project report

SCE12-0548

Visual Event Recognition

Supervisor: A/P Xu Dong

Examiner: A/P Cham Tat Jen

Author: Gong Li

Submitted in Partial Fulfillment of the Requirements
for the Degree of Computer Engineering
of Nanyang Technological University

School of Computer Engineering
November 2013

Abstract

This report summarizes the work that has been done in the final year project of recognizing visual events in videos. It starts with image recognition, in which images are represented in spatial pyramids. Such representations are then input into SVM and KNN for recognition. In video recognition, bag of words and specialized Gaussian Mixture Models are employed to represent videos, and respective distance calculation is used to measure video-to-video distance. These distance matrices are then input into SVM for recognition using different kernel types. Also, four domain adaptation methods are implemented to recognize Kodak consumer videos using Youtube videos. Adaptive multiple kernel learning achieves the best and improves the mean average precision from 44.33% to 61.40%. Last but not least, a web-based demo system is implemented in two modes to visually demonstrate the underlying recognition system.

Acknowledgements

I would like to express my sincerest gratitude to to the following people for their guidance and support towards the accomplishment of this final year project.

Associate Professor Xu Dong, who is my supervisor, guided me during the whole period of this project. His comments, feedbacks and encouragement have greatly helped me accomplish this project. Also, his opinion on the difference between academia and industry shaped my mind and helped me understand myself better.

Mr. Li Wen and **Mr. Chen Lin**, who are both Ph.D. candidates, shared their experience on various approaches to improve the recognition performance. Furthermore, I would like to thank them for sharing Youtube videos with me.

Ms. Chua Poo Hua and **Ms. Siom Siew Ling**, who are both Laboratory Executives, helped me to set up the workstation and installed all necessary software.

Lastly, I would like to thank all my friends for spending time with me. It is feeling so good to meet all of you.

Thank you all.

Contents

List of Tables

List of Figures

1 Introduction

Nowadays, with the development of technology, digital cameras and mobile phones with cameras are becoming more and more common. As a result, the number of videos is increasing exponentially. According to Youtube statistics, 100 hours of videos are uploaded every minute, and over 6 billion hours of videos are watched each month in Youtube. Due to such a large number of videos, it is urgent and important to develop effective and robust approach to index and retrieve videos. Also, because of the large number of videos, it is costly to rely on human power to recognize videos. Thus, one direction is to recognize videos by leveraging machine intelligence. The general way is to use a set of training videos with labels to build classifiers. Afterward, these classifiers could be used to predict labels for new coming videos. However, in order to build a robust classifier, there is a lot of work to do.

In this final year project, various approaches are studied and implemented to recognize videos. This project starts with image recognition because the data size of images is much smaller compared with videos. To represent images, SIFT features [25] are extracted from images, the features of each image are converted into a histogram based on visual vocabulary built through clustering algorithm using spatial pyramid matching [19]. Then such representations are input into machine learning classification methods including SVM and KNN for recognition. Furthermore, to better calculate image-to-image distance, earth mover's distance (EMD) [33] is incorporated, and the recognition accuracy demonstrates the effectiveness of EMD. Also, a method to identify representative images from a cluster of images is introduced, and this method is later on use to compress videos to remove redundant frames. This project then moves to video recognition, in which two types of representations are adopted to represent videos. The first one is the famous bag of words model, which treats each video as multiple frames, and represents each video as a stack of histograms [11] where each histogram represents a frame in that video. Moreover, in the second representation, each video is treated as a bag of SIFT features [45], and specialized Gaussian Mixture Models (GMM) are built on this bag to represent this video. Before building specialized GMM, a global GMM is first built on SIFT features of all training videos. Due to two different representations of videos, two different distance calculations are used. Furthermore, concept attribute [22], which models each video as a vector of scores of semantic concepts, is also studied and implemented. Based on the performance, it shows that such short vector representation still retains comparable discrim-

inable power compared with original representations of videos. This is not the end of story. Promising domain adaptations methods [11, 10, 43, 9], which leverage labelled samples from other domains, are also researched. Feature Replication, Adaptive SVM, Domain Transfer SVM and Adaptive Multiple Kernel Learning are implemented in Python and improve the performance of recognition from the original 44.33% to as high as 61.40% using the metric of mean average precision. Lastly, with popular open source packages including Django and Bootstrap, a web based demo system is implemented to present the work of this project.

The rest of this report is organized as follows: firstly, literature review is presented followed by the section of details on image recognition. Secondly, the section of video recognition, which introduces Aligned Space-Time Pyramid Matching and Specialized Gaussian Mixture Models, is presented. Thirdly, four domain adaptations methods are talked about followed by the section of comprehensive experiments. Next, the design of web-based demo system is briefly introduced. Finally, this report ends with the section of conclusion.

2 Literature Review

This section introduces related work on visual events recognition. There are roughly three important aspects in video recognition. The first one is to extract features from videos. Various features have been invented and improved by researchers over decades. The second aspect is the model used to represent videos. Once features are extracted, they are still not ideal for recognition. As a result, more appropriate models are needed to compactly represent videos. Lastly, there are various classification approaches. In the rest of this section, details of the above aspects will be illustrated.

For feature extraction, it depends on how videos are viewed. The first one is that video could be treated a series of consecutive frames. As a result, those features in images could be adopted. The features of images could be classified into two categories: local features and global features. Local features refer to descriptions of interest points. The most widely used one is Scale Invariant Feature Transform [25], which describes interest points where the centers differ from their neighbors using Difference-of-Gaussian. Other descriptors include Histogram of oriented gradients (HOG) [8] and Local binary pattern (LBP) [30]. Global features refer to those which encode the image based on the overall distribution of color, texture and edges. Some popular global features include color histogram and Gabor texture [26]. However, if video is put onto spatial-temporal dimension, where the time dimension is also considered, there are some other features to extract. Laptev used Harris corner patch detector [21] to locate spatial-temporal interest points (STIP) and adopted the concatenation of Histogram of Oriented Gradient (HOG) and Histogram of Optical Flow (HOF) as descriptor [17]. Furthermore, acoustic information is also an important part in videos. Features like MFCC [2], which represents the short-term power spectrum in audio signals, are also often used in video event recognition. Furthermore, given the above various features, it is also possible to combine them in order to comprehensively describe videos.

Once features are extracted, a common framework called as Bag of Words [36], which treats images as "documents" and uses histograms to represent images using visual vocabulary, is generally employed to convert frames into histograms. Because one image counts for one histogram, the video is then converted into a stack of histograms. Beyond bag of words, Grauman et al. [12] proposed *pyramid matching*, and Lazebnik extended this idea to spatial pyramid matching [19] which takes spatial information into consideration. Apart from BoW, Zhou et al.

[45] proposed to represent videos as Gaussian Mixture Models. Firstly, a global GMM is built on SIFT features of training data. Secondly, specialized GMMs are built for video clips based on this global GMM in Maximum a Posteriori way. Each video clip is then represented by a fixed-length vector of means of Gaussian components.

As for recognition methods, Support Vector Machine (SVM) [5] is currently the most popular classification method in event recognition systems. One fantastic feature of SVM is that it could project samples onto higher dimensional space through kernel functions easily. To make SVM better, many researchers proposed domain adaptation methods [9, 43, 10, 11], which leverage labelled samples from other domains. Normally, these methods update the objective function and maximization function of SVM to take training samples from other domains into consideration. Furthermore, graphical models include Hidden Markov Model (HMM) and Bayesian network (BN) are also employed for recognition. Xie et al. [40] proposed to use multiple levels of HMM to classify play and non-play segments of soccer videos. Moreover, Huang et al. [13] proposed to use dynamic Bayesian network to perform semantic analysis on soccer videos.

To conclude, there are three aspects in video recognition: feature, model and classification. Various approaches on these three aspects have been proposed. Those approaches introduced in literatures have laid a solid foundation in the filed of video recognition. Because of the foundation, we can see further by standing on the shoulders of giants. In other words, this project could never be done well without researchers' efforts.

3 Image Recognition

Image recognition is a very interesting topic in computer vision and pattern recognition, and the goal is to recognize different classes in images. For instance, there are trees, people, sun, moon and buildings in different images. Human can easily distinguish different classes among a set of images. However, machines are not good at such tasks. For many years, a lot of researchers are working on this topic and have achieved significant results.

So far, a very good framework to recognize images is the so-called *bag of words* model, which generalizes from natural language processing (NLP). In NLP, bag of words basically aims to represent an article using histogram of word frequencies. Following this idea, in image recognition, an image is also represented as a histogram of visual words, and these visual words normally come from centroids generated from clustering algorithms. Later on, such representations of images are used to build up generic machine learning models, and the accuracy is quite impressive. During recognition phase, the most critical process is to calculate image-to-image distance. One direct way is the Euclidean distance. But a better way to calculate distance is called as Earth Mover's Distance [33], which finds the best matches between image parts and produces a more accurate distance.

In the following of this section, an image recognition framework consisting of 4 steps is first introduced. Secondly, Earth Mover's distance and its applications in image recognition is talked about. Lastly, the method to identify duplicates in a cluster of images used in this project are presented, and this method is going to be applied in video recognition to show the effectiveness of Earth Mover's distance.

3.1 Framework Description

Generally, there are 4 below steps to go:

3.1.1 Extract SIFT features from each image

It is almost impossible to directly rely on pixel data of images to perform recognition tasks, and the main reason is because such pixel data does not possess much discriminative power. Therefore, other approaches to represent images are needed. One approach is to extract features from images and use those features to represent images compactly. There are generally two categories of features: global features and local features. As the name stated, global features encode a

whole image based on the overall distribution of color, texture, or edge information. Some popular global features include color histogram and Gabor texture [26]. On the other hand, local features refer to features extracted from local patches (or interest points). There are two steps involved in extracting local features: detection and description. Detection refers to localizing interest points in images. Among popular detection algorithms, the most widely used is Difference-of-Gaussian (DOG) [25] introduced by David Lowe, which detects blob regions where the centroids differs from their neighbors. Other popular detectors include Harris-Laplace [21], Hessian [27], etc. The next stage is description which aims to describe those located interest points in meaningful manners such that this descriptor is invariant to scale, viewpoint, rotation and illumination changes. There are lots of descriptors that have been designed over years. The best known shall be Scale Invariant Feature Transform (SIFT) [25], which concatenates histograms of gradient orientation around interest points. Other descriptors include Histogram of oriented gradients (HOG) [8], Local binary pattern (LBP) [30], etc. Given the robustness of SIFT shown in various literatures [11, 45, 19, 25], SIFT is chosen to be the feature to be extracted from images through the whole period of this project.

Now it's time to briefly elaborate details about SIFT. There are two steps when performing extracting SIFT features. The first step is to identify interest points, and the criteria is based on *difference-of-Gaussian functions*:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{1}$$

where G is the Gaussian 2D kernel, $I(x, y)$ is an input image and k is a constant multiplicative factor which separates two nearby scales. After computing $D(x, y, \sigma)$ for all sample points, those sample points with local maxima and minima of $D(x, y, \sigma)$ compared with their neighbors are selected to be interest points. In order to better localize interest points, these candidate locations are filtered to remove unstable points based on criteria including low contrast and edge response. Please refer to the paper [25] for details on removing unstable points. The next step is to compute a descriptor based on those interest points located in previous step. For each interest point, a descriptor is created by first computing the gradient magnitude and orientation in a grid of subregions around that specific interest point. These gradient orientations are then accumulated into orientation histograms. Finally, these histograms are concatenated to form a descriptor vector. The recommended setting uses 4×4 subregions with 8 bin

histogram, which results in a 128 bin histograms ($4 \times 4 \times 8 = 128$). Figure 1 below demonstrates the whole process to construct SIFT descriptor.

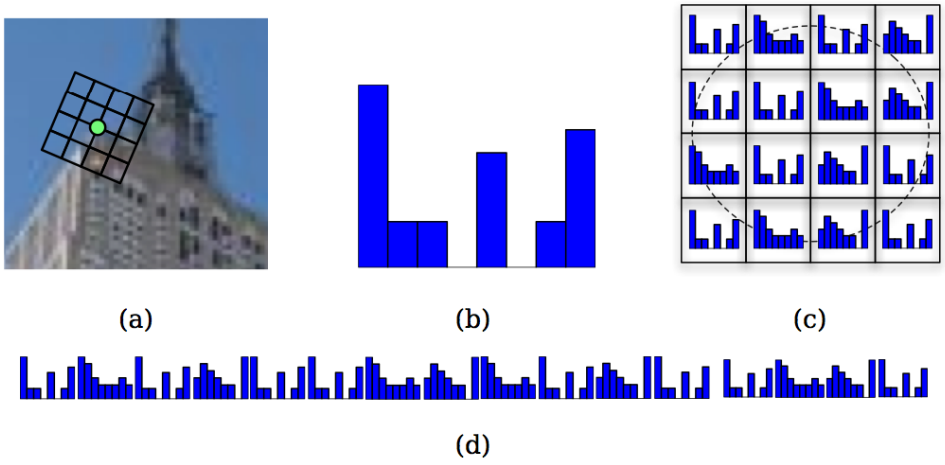

Figure 1: SIFT descriptor [37] (a) a frame with 4×4 subregions around an interest point. (b) an 8 bin histogram over the direction of gradient in one subregion. (c) all histograms in their respective subregions. (d) all 16 histograms are concatenated to form a 128 (16×8) dimensional vector.

For simplicity, an open source library VLFEAT [38] is adopted to extract SIFT from images. A typical image of size 200×200 normally produces around 800 SIFT features.

3.1.2 Build up visual vocabulary from features of training set

Although SIFT features have been extracted, these 128 dimensional raw features are still not an ideal representation because of limitations including the curse of dimensionality. Thus, a visual vocabulary is needed to map high dimensional descriptors to words by quantizing the feature space. Commonly, vocabulary is built through applying K-Means [23] clustering algorithm on available features, and the final centroids output by K-Means are treated as representative words. The basic K-means algorithm goes as below:

Algorithm 1 Basic K-means Algorithm

1: Randomly select K points as the initial centroids
2: **do**
3: Form K clusters by assigning all points to their closest centroids.
4: Recompute the centroid of each cluster
5: **while** Centroids are changed

Based on the above algorithm, it is easy to see that the time complexity is $O(n*K*I*d)$, where n is the number of points, K is the number of clusters, I is the number of iterations and d is the number of dimensions. This classical K-means algorithm works fine for small data sets but is very expensive for large data sets. Later on, vocabulary is also needed for video recognition where the number of features is nearly 0.4 million after sampling. By noticing that the available computational resource is limited, it is not affordable to apply classical K-Means on large data sets. Therefore, a better method is needed to build up vocabularies. Mini-Batch K-Means [35], a variant of KMeans algorithm which uses mini-batches to reduce the computation time, is a good alternative. In contrast to other algorithms which reduce the convergence time of k-means, mini-batch k-means produces results that are generally only slightly worse than the standard algorithm. The details about mini-batch k-means are presented as follows:

Algorithm 2 Mini-batch K-Means

Given: k, mini-batch size b, iteration t, data set X
Initialize each $c \in C$ with an x picked randomly from X
$v \leftarrow 0$
for $i = 1$ to t **do**
    $M \leftarrow b$ examples picked randomly from X
    for $x \in M$ **do**
        $d[x] \leftarrow f(C, x)$ ▷ Cache the nearest center to x
    end for
    for $x \in M$ **do**
        $c \leftarrow d[x]$ ▷ Get cached center for this x
        $v[c] \leftarrow v[c] + 1$ ▷ Update per-center counts
        $\eta \leftarrow \frac{1}{v[c]}$ ▷ Get per-center learning rate
        $c \leftarrow (1 - \eta)c + \eta x$ ▷ Take gradient step
    end for
end for

Once this vocabulary is built, a histogram whose length equals to the size of vocabulary is built for each image. Afterward, features of images are examined to construct their histograms. For each SIFT feature, the nearest word to this feature is found, and the bin representing the found word is increased by one. After finishing examining all features, a histogram is constructed and is used to represent the respective image. As a result, all images are transformed from pixels to fixed length histograms. In image recognition experiments, the number of centroids is set to be 300 when performing Mini-Batch K-Means. Thus, all images are converted into 300 dimensional histograms.

3.1.3 Construct a pyramid of three levels for each image

Now that histograms for images are built, these histograms could be put into machine learning models including K Nearest Neighbor and Support Vector Machine for recognition. However, this is not the end of story. To enable more discriminative power, Grauman and Darrell [12] proposed *pyramid matching* in 2005 to find an approximate correspondence between 2 sets of data. The basic idea is to map sets of features to multi-resolution histograms, and then compare the histograms with a weighted histogram intersection measure in order to better approximate the similarity of best partial matching. As reported in the paper [12], this pyramid match kernel produces much better recognition results for similar running time.

Though *pyramid matching* fuses information from multiple levels built on different resolutions, this method ignores spatial information because features are matched without any order. To partially address this problem, in the following year, Lazebnik introduced spatial pyramid matching [19]. Unlike pyramid matching which varies histogram matching resolution, spatial pyramid matching varies the spatial resolution at which they are aggregated. Figure 2 below demonstrates the case when there are three levels in spatial pyramid matching. The three levels are presented from the left to right. Level 0 is simply a standard bag of features which treats the whole image as a whole. For level 1, this image is divided equally into 4 parts, each part contributes to one histogram built on the same vocabulary. Furthermore, level 2 is the case when image is divided into 16 parts which results in 16 different histograms. Finally, all these histograms are concatenated with appropriate weights to form a "long" histogram with dimensionality as $M \sum_{l=0}^{L} 4^l$, where M is the size of vocabulary and L equals to total number of levels -1. For instance, the vocabulary size used in experiments is 300. Thus, the number of dimensions is 1500 for two-level spatial pyramid and 6300 for three-level spatial pyramid. Additionally, the weights for histograms at different levels are listed below:

$$weight(l) = \begin{cases} \frac{1}{2^L} & if\ l = 0 \\ \frac{1}{2^{L-l+1}} & Otherwise \end{cases} \tag{2}$$

For example, as shown in Figure 2, weights for three levels histograms are $\frac{1}{4}$, $\frac{1}{4}$ and $\frac{1}{2}$, respectively. It is easy to see that such weighting scheme tends to penalize matches found in larger cells, and the reason why it makes sense is because larger cells allow more dissimilar features.

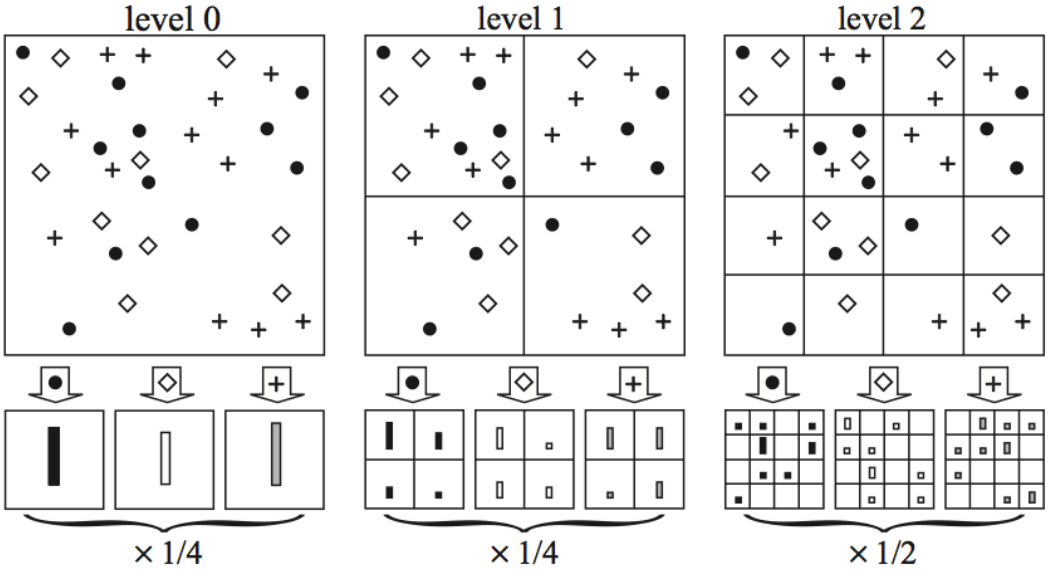

Figure 2: Example to construct a three-level pyramid [19]. At top, images are divided at three different levels of resolution. Next, the count to each feature is calculated for each spatial bin. Finally, each histogram is weighted according to equation (2)

3.1.4 Classify based on above representations

In machine learning, classification refers to the problem of identifying to which category a new observation belongs, on the basis of training data set. Generally, machine learning builds a model which fits the training data set. Afterward, this model to use to recognize new data set. There are many classification methods including K-nearest neighbors (KNN), support vector machine (SVM), decision tree, etc. In this project, KNN and SVM are incorporated into the recognition system, and details about these two kinds of classification methods will be presented in following paragraphs.

K-nearest neighbor (KNN) K-nearest neighbor is considered as a lazy learning algorithm. This is because it does not attempt to construct a general internal model, but just store all instances of the training data. Once a test case is input, KNN retrieves the nearest $K(K > 0)$ neighbors to this test case from training data, and the predicted label is computed based on labels of neighbors in a majority vote manner. In other words, the predicted label of a test case is chosen to be the most representative label among K nearest neighbors. For example, Figure 3 shows how to predict label for x_q when K is set to be 5. Among 5 neighbors of x_q, three are negative while the left two are positive. Therefore, x_q is predicted to be within negative class based on majority vote.

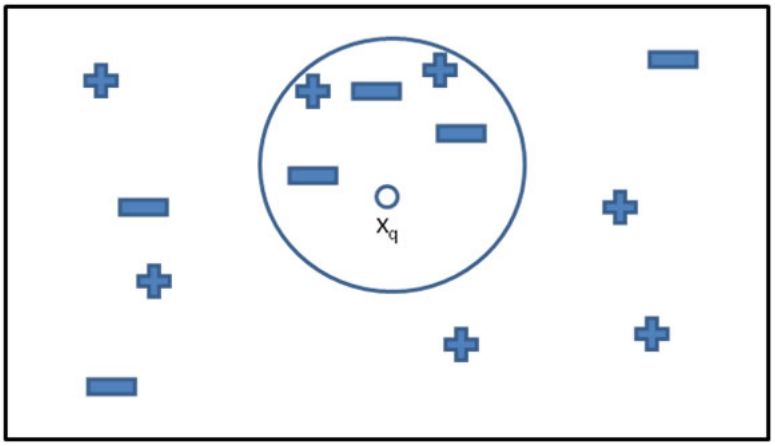

Figure 3: A case of k nearest neighbor. Among 5 neighbors of x_q, three are negative while the other two are positive. As a result, the label of x_q is predicted to be negative.

Support vector machine (SVM) Support Vector Machine was introduced in COLT-92 by Boser, Guyon & Vapnik [5] and became rather popular since then. This is a well motivated algorithm which is developed from statistical learning theory since 1960s. It has been successfully applied into many fields (bioinformatics, text, image recognition, ...) and is currently the most popular classifier in event recognition systems [14].

SVM starts with the problem of finding the optimal separating hyperplane which separates a linearly separable dataset:

$$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$$

where $x \in R^D$ and $y \in \{-1, +1\}$. An example is presented in Figure 4. Black solid circle represents one class while the while circle represents another class. The solid line $\mathbf{wx} + b = 0$ is the optimal hyperplane, and the data points on $\mathbf{wx} + b = 1$ and $\mathbf{wx} + b = -1$ are called are *support vectors*. It is not difficult to see that the distances from support vectors to optimal hyperplane are maximized. Therefore, the finding of optimal hyperplane is transformed into an optimization problem:

$$\begin{aligned} \text{minimize} \quad & J(w) = \tfrac{1}{2}\|w\|^2 \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \quad \forall i \end{aligned} \tag{3}$$

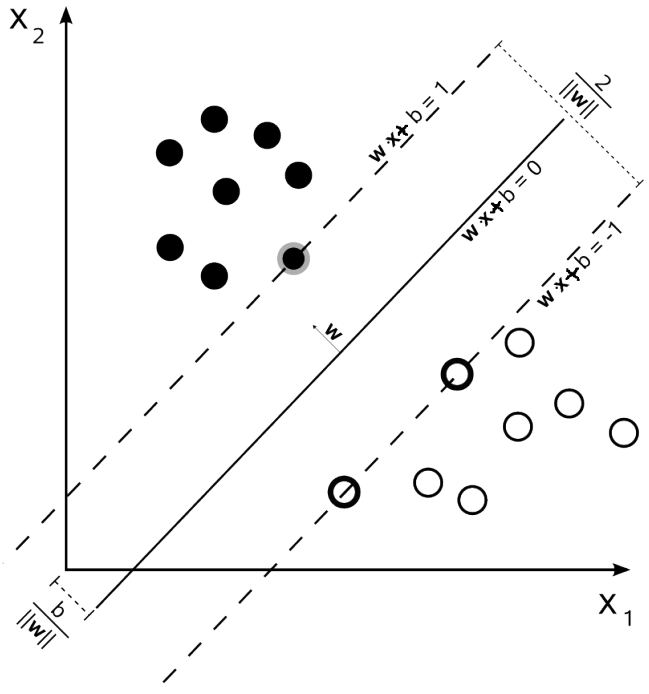

Figure 4: Optimal hyperplane which separates a linearly separable dataset

This optimization is generally solved by introducing the Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w^T x_i + b) - 1] \tag{4}$$

where α_i are dual variables. After substituting stationary conditions, the Lagrangian gives the dual cost function:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j x_i^T x_j \tag{5}$$

The optimization is now:

$$\begin{gathered} \hat{\alpha} = \arg \max_{\alpha} W(\alpha) \\ \text{subject to} \quad \alpha_i \geq 0 \quad \text{and} \quad \textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \end{gathered} \tag{6}$$

So far, the problem of finding s saddle point for $L(w, b)$ becomes an easier one of maximizing $W(\alpha)$. Once this optimization problem is solved, the decision function to predict new data points is shown as follows:

$$f(x) = \text{sign}\left(\sum_i y_i \alpha_i x_i^T x + b \right) \tag{7}$$

Please note that in equation (7), the point set $\{x_i\}$ includes all calculated support vectors. Moreover, the training data appears as dot products $x_i^T x_j$ in $W(\alpha)$, and it means that classification could be done in higher or even infinite dimensional space by changing $x_i^T x_j$ into other forms of kernel function. Suppose there is a kernel function $K(x, y)$, the original optimization problem is converted by replacing $x_i^T x_j$ with $K(x_i, x_j)$:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{8}$$

Because this transformation has nothing to do with α_i and y_i, the optimization remains the same:

$$\begin{aligned} \hat{\alpha} &= \arg\max_{\alpha} W(\alpha) \\ \text{subject to} \quad \alpha_i \geq 0 \quad &\text{and} \quad \textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned} \tag{9}$$

Finally, the decision function becomes:

$$f(x) = \text{sign}\left(\sum_i y_i \alpha_i K(x_i, x) + b \right) \tag{10}$$

As shown above, thanks to the kernel function $K(x_i, x_j)$, nonlinear classification is achieved without projecting data points into higher dimensional space. This clever trick refers to "kernel trick". Detailed information can be found at [6].

There are many kernels designed over years, and different kernels have different impacts on recognition performances. In experiments of image recognition, the below four kernel functions have been implemented.

Kernel type	**Kernel function**
Linear	$x_i^T x_j$
Polynomial	$(\gamma x_i^T x_j + r)^d$
RBF	$\exp(-\gamma \|x_i - x_j\|^2)$
Sigmoid	$\tanh(\gamma x_i^T x_j + r)$
Histogram intersection [3]	$\sum_{p=1}^{D} min(x_i^p, x_j^p)$

Table 1: Kernel functions

Although the above mathematical description of support vector machine seems to be complex, there are lots of open source libraries of SVM. One famous and popular package is LIBSVM [7]. What's better is that sklearn [31] provides well

defined python wrapper to invoke LIBSVM. All the work facilitates the use of SVM, and it demonstrates the collaboration among researchers all around the world.

3.2 Earth Mover's Distance (EMD)

In this subsection, Earth mover' distance will be introduced, and the reason is because later on in video recognition, this kind of distance calculation is going to be heavily used. The rest of this section is organized as follows. Firstly, the mathematical model of earth mover proposed by Rubner, Tomasi and Guibas [33] is presented followed by an example of calculating earth mover distance between two images. Last but not least, experiments have been done to show the effectiveness of EMD, and the result is recorded in the section of experiments.

The earth mover's distance is formulated as the following linear programming problem: Let $P = \{(p_1, w_{p_1}), ..., (p_m, w_{p_m})\}$ be the first signature with m clusters, where p_i is the cluster representative and w_{pi} is the weight of the cluster; $Q = \{(q_1, w_{q_1}), ..., (q_n, w_{q_n})\}$ the second signature with n clusters; and $D = [d_{ij}]$ the ground distance matrix where d_{ij} is the ground distance between clusters p_i and q_j. We want to find a flow $F = [f_{ij}]$, with f_{ij} being the flow between p_i and q_j, that minimizes the overall cost:

$$\begin{aligned} \text{minimize} \quad & Work(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \\ \text{subjective to} \quad & f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \\ & \sum_{j=1}^{n} f_{ij} \leq w_{p_i} \quad 1 \leq i \leq m \\ & \sum_{i=1}^{m} f_{ij} \leq w_{q_j} \quad 1 \leq j \leq n \\ & \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}) \end{aligned} \tag{11}$$

Once the above linear programming problem is solved, the earth mover's distance is defined as the resulting work normalized by the total flow:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \tag{12}$$

To better understand EMD, here is an example to incorporate EMD into the distance calculation of images. To do so, the first step is to divide each image into four parts equally. An example is depicted in Figure 5. There are two images A and B, and they are both divided into 4 parts equally. Each part is

Figure 5: Image A and B are divided into 4 parts equally. Then each part is converted into a histogram. Thus, four histograms are stacked to represent each image.

then represented by a histogram based on previous built vocabulary. It is easy to see that this division is identical to level 1 in spatial pyramid matching, which also divides an image into 4 equal parts. Now image A and B are represented by 4 histograms each. After adding equal signature value to these histograms, A and B are represented as follows:

$$A = \{(a_0, 0.25), (a_1, 0.25), (a_2, 0.25), (a_3, 0.25)\}$$

$$B = \{(b_0, 0.25), (b_1, 0.25), (b_2, 0.25), (b_3, 0.25)\}$$

where a_0, a_1, a_2, a_3 and b_0, b_1, b_2, b_3 are histograms generated from respective parts.

The next steps is to calculate the distance matrix D with each cell d_{ij} representing the distance between a_i and b_j. In this case, Euclidean distance is adopted, and the calculated distance matrix is shown below in Table 2.

	b_0	b_1	b_2	b_3
a_0	27.06473721	23.37733946	29.18047292	32.61901286
a_1	20.84466359	21.50581317	21.70253441	30.34798181
a_2	26.48584528	27.38612788	27.46816339	34.71310992
a_3	19.31320792	24.20743687	21.59861107	31.81194744

Table 2: Distance matrix between image A and B

If EMD is not used meaning that there is no alignment of image segments, a_0 is matched to b_0, a_1 is matched to b_1, a_2 is matched to b_2 and a_3 is matched to b_3. As a result, distance without alignment is calculated as below:

$$distance_{unaligned} = d_{00} + d_{11} + d_{22} + d_{33} = 107.851$$

However, if EMD is used, the best matches between image segments will be found after the linear programming problem in EMD is solved. For image A and B, the best matches found are shown in Figure 6. Compared with unaligned case, this one matches the gun and human body properly. Following the matches in Figure 6, distance with alignment is calculated as below:

$$distance_{aligned} = d_{30} + d_{12} + d_{01} + d_{23} = 99.106$$

Figure 6: Matches between segments of A and B calculated by EMD. As we can see, the gun, human body and ground are matched properly. Therefore, distance with alignment shall be more accurate.

Now that the calculation of distances between images has been defined, it is needed to select a kernel type for SVM to recognize. Apart from RBF (also called as Gaussian kernel) introduced before, three more kernel types were tried in experiments, and they are listed in Table 3. Details of performances of these four kernel types are presented in the experiment section.

Kernel type	**Kernel function**
Gaussian	$\exp(-\gamma D^2(I_i, I_j)$
Laplacian	$\exp(-\sqrt{\gamma} D(I_i, I_j)$
ISD	$\frac{1}{\gamma D^2(I_i, I_j)+1}$
ID	$\frac{1}{\sqrt{\gamma} D(I_i, I_j)+1}$

Table 3: Four kernel types which have been tested. $D(I_i, I_j)$ represents the distance between I_i and I_j. The kernel parameter γ is selected to be the default one $\gamma = \frac{1}{A}$, where A is the mean value of the squared distances between training samples as suggested in [18].

3.3 Key Frame Identification in a Cluster of Images

A video could be simplified as a cluster of consecutive frames. By noticing that some consecutive frames in video are similar to each other, it is thought that video can be described by several key frames. In doing so, the data to represent each video can thus be largely reduced. One key step in key frame identification is to check whether two images are near duplicate. If two or more images are examined to be near duplicate to each other, only one image will be selected to represent these images. The rest of this section will focus on two parts: near duplicate identification and how to apply near duplicate identification into key frame identification in videos.

3.3.1 Identifying near duplicate

The way which has been implemented to check whether two images are near duplicate follows the method stated on paper [44]. In this paper, there are basically three steps to check whether two images are near duplicates. Firstly, the authors propose to use a hash table to match SIFT points. Secondly, a SVM classifier is built based on the matching SIFT points. Finally, this classifier could be used to check near duplicates. Due to a lack of training set, the classifier is not built and thus has to be replaced by threshold checking. In other words, two images are treated as near duplicates as long as the number of matching interest point is large enough compared with a predefined threshold.

In order to minimize false matches, the authors propose one-to-one symmetric matching [44], which ensures all the matches are nearest neighbors. One the other hand, the symmetric property makes sure that the matching result of set A to B is exactly the same as B to A. Suppose there are two images I_1 and I_2, the steps to check whether these two images are near duplicate go as below:

1. **Perform PCA on SIFT features of I_1 and I_2 to reduce dimensions**
   The dimension of SIFT feature is reduced from 128 to 36, and each value of reduced feature is normalized to be within the range [0, 2].

2. **Hash all interest points of I_1 into a 8×36 table**

   The hash table is composed of 8×36 bins as shown in Figure 7. Given each point $P = [p_1, p_2, ..., p_{36}]$ of I_1, the index of p_i is hashed to,

$$H(p_i) = \lfloor p_i \times 4 \rfloor$$

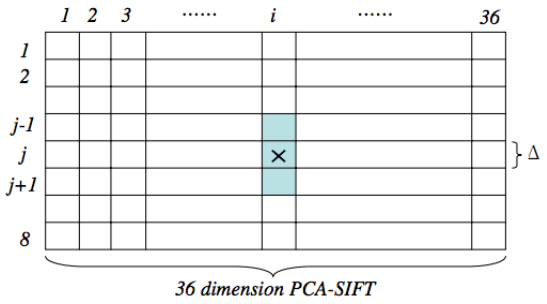

Figure 7: 8×36 Hash Table [44]

Since the dimension is 36, P is repeatedly indexed into the corresponding bin for 36 times, according to its quantized value in a particular dimension.

3. **For each interest point Q of I_2, examine whether there is a match in I_1**
   There are three sub steps to go:

   (a) Hash Q into the hash table

   (b) Retrieve the set $A(Q)$ satisfying the below constraints
   For each interest point $P \in I_1$, put P into $A(Q)$ if

$$\sum_{i=1}^{36} f(q_i, p_i) = 36$$

   where

$$f(q_i, p_i) = \begin{cases} 1 & \text{if } |H(q_i) - H(p_i)| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

   (c) If $A(Q)$ is not empty, find the nearest neighbor $M \in A(Q)$ with one-to-one symmetric constraint as Q' match

4. **If the number of matching interest point is large enough, I_1 and I_2 are near duplicates**

An example of performing the above algorithm is depicted below in Figure 8. Those colored lines in the upper two images connect the matched interest points, and it is easy to distinguish that the matching accuracy is quite high because of one-to-one constraint [44]. Please also notice that only partial matching points are drawn for better visualization.

Figure 8: Near duplicate example. The colored lines in upper two images connect partial matched interest points, and the below two images are original images.

3.3.2 Identifying key frames

Now that near duplicate method is introduced, it is time to check how this method could be incorporated into key frame identification. The naive brute-force identification checks any two frames in a video, and the computational cost is $O(cn^2)$, where c is the cost of identification and n is the number of all frames. Because c is almost fixed, the left way to reduce computational time is to reduce n. Inspired by the paper [39], it is recommended to perform K-means algorithm on all the frames at first. Later on, the previous introduced near duplicate identification is performed within each cluster. In this case, if all clusters have equal size, the cost becomes $O(cn^2/r)$, where r is the number of clusters.

Once the clusters are calculated through K-means, the steps to identify key frames in each cluster go as below:

1. **Build a graph for each cluster**
   Each image in that cluster is treated as a node. If two images are identified as near duplicates, an edge is established between these two image nodes. Once all combinations are processed, a graph is built.

2. **Choose representative nodes from the graph**
   The first thing to do is to check whether there are connected components

in the graph. For each connected component, the node with the largest number of edge is selected to be a key frame. If there is a tie, the key frame is then randomly chosen among candidates.

A very good example is illustrated in Figure 9. These frames are sampled at a rate of one frame per second from a video introducing how to use Google glass from Youtube. Figure 9 depicts how a cluster is processed to produce key frames. There are three connected components inside this cluster. Next, one key frame is extracted from each connected component, and all the three resulting frames are treated as three key frames of this cluster. Because all similar frames are discarded, these three resulting frames are indeed representative.

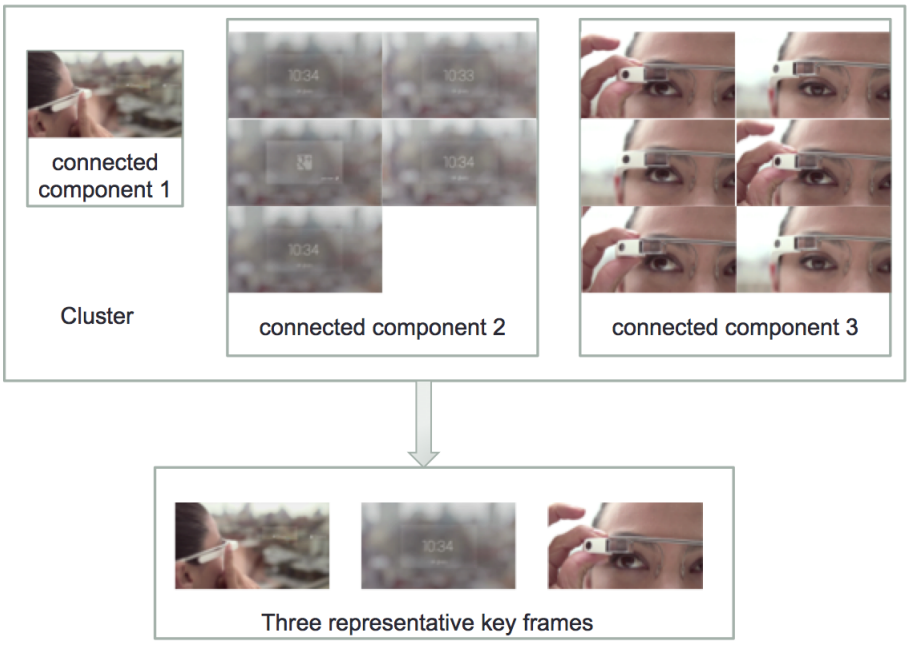

Figure 9: Key frames identification example. In this cluster, there are three connected components. One frame is selected from each connected component. The resulting three frames are treated as representative frames.

By combining all representative frames from all clusters, a video could be represented by those key frames. Therefore, the size of video is decreased to some degree and thus accelerates the process of recognition.

4 Video Recognition

As introduced in the section of image recognition, recognition tasks could be simplified into a problem of calculating distances between different samples in the evaluated data set. This is because once the distance matrix is calculated, this distance matrix could be input into K-nearest neighbor algorithms or better approaches like Support Vector Machine for recognition. However, it is not so easy to compare two raw videos in quantitative way and thus difficult to calculate distances using raw video data. In order to resolve this problem, a compact representation similar to image for each video clip is needed.

The rest of this section is organized as follows. Firstly, two different representations of video are introduced followed by the distance calculation for each of these two representations. Once the task of calculating distances is finished, four types of kernels used in image recognition are tested. Finally, another method using concept attributes is presented.

4.1 Representations of Videos

4.1.1 Bag of words (BoW)

Naive BoW Video consists of a series of consecutive frames and generally presents frames at rates ranging from 20 to 50 frames per second. To be simple, frames sampled from a video every one second could be used to represent this video. For instance, a two-minute video is then represented by 120 frames. With the same representation of image illustrated in section of image recognition, these 120 frames are then converted into 120 histograms. At last, these 120 histograms are stacked to represent this two-minute video.

A formal description of this approach to build bag of words model for videos go as below.

1. **Build vocabulary**
   Similar to image recognition, a vocabulary is needed to represent each frame of videos as a histogram. The vocabulary is built by applying Mini-Batch K-Means [35] clustering algorithm on SIFT features of sampled frames for all available videos. In the experiments, two vocabularies were built with the numbers of centroids being 2500 and 1000.

2. **Represent each video as a stack of histograms**

Let's say the number of sampled frame in a video is M, and each video is represented as a $1 \times V$ histogram, where V is the size of vocabulary. Then histograms of these frames from this video are stacked together to form a $M \times V$ matrix, and this matrix represents this video.

Following the above steps, videos are converted into a compact matrix with each row representing a frame. However, different videos are converted into matrices with different rows because of different durations. Such differences make it not easy to use general distance calculation formula to calculate video-to-video distance. The solution to this problem will be presented in next section about distance calculation.

Better BoW In naive Bow, SIFT feature are assigned to its nearest word, and the respective bin in histogram is increased by one. Two questions arise from this statement. Why SIFT feature can only be assigned to one word but not multiple words? Why the respective bin is increased by one but not some other values? To answer these two problems, soft assignment and different weighting schemes are introduced.

- Soft assignments
  Soft assignment allows that a visual feature could be assigned to multiple words rather than only one word. In doing so, more valuable information is retained during quantization process and thus might provide more discriminative power. A straightforward approach [15] is that the top N nearest words are selected for each visual feature. Let's say the size of vocabulary is K, and thus a K-dimensional vector $T = [t_1, t_2, ..., t_K]$ is used to represent an image. The algorithm to construct this vector goes as below.

  The above algorithm selects the top N nearest neighbors. What if N equals to the size of vocabulary? Here, instead of the original algorithm, Agaral and Triggs [1] proposed to use Gaussian mixture model (GMM) built from training data to perform assignment. Let's say the number of components in Gaussian mixture model is K. For each visual feature, GMM produces a K-dimensional vector representing posterior mixture-component membership probabilities. Finally, all these K-dimensional vectors are summed up to produce one final K-dimensional vector to represent the respective image.

- Weighting schemes
  In previous implementation of constructing histograms, only term (word)

Algorithm 3 Build histogram with soft assignment

Given: vocabulary size K, words in vocabulary $[w_1, w2, ..., w_k]$, visual features F, parameter N
Initialize a K-dimensional vector $T = [t_1, t_2, ..., t_K]$ with all components $t_i = 0$

for $f \in F$ **do**
    Retrieve the top N nearest words to f
    Put the indexes of the top N words in W_N in sorted distances order
    $v \leftarrow 0$
    for $index \in W_N$ **do**
        $t_{index} \leftarrow t_{index} + \frac{1}{2^v}$
        $v \leftarrow v + 1$
    end for
end for

return T

frequency is taken into consideration. Such implementation ignores global information because the frequencies of words in the global corpus are not considered at all. To address this problem, *inverse document frequency(idf)* has already been proposed in information retrieval and becomes rather important [34]. *Inverse document frequency* of visual word t_i is defined as follows.

$$idf(t_i) = \log(N/n_i) \tag{13}$$

where N is the total number of images in the corpus, and n_i is the number of images having visual word t_i. According to this formula, the more frequent a visual word is, the smaller *idf* is. Therefore, if $tf_i \cdot idf_i$ is used to replace tf_i, the weights of words which occur frequently are diminished while the weights of words which occur rarely are increased. Another factor is whether to normalize the vector or not. By taking all these factors into consideration, different weighting schemes are summarized in the below Table 4. Experimental results of these weighting schemes are demonstrated in the section of experiments.

4.1.2 Gaussian mixture models

In the previous section which talks about employing bag of word model to represent videos, one approach to perform soft assignment is achieved through Gaussian mixture models (GMM). Actually, there are more things that GMM could do. Zhou et al. [45] proposed to represent videos as specialized GMM, which

Name	Factors	Value for t_i
bxx	$binary$	1 if t_i presents, 0 if not
txx	tf	tf_i
txc	$tf, normalization$	$\frac{tf_i}{\sum_i tf_i}$
tfx	tf, idf	$tf_i \cdot \log(N/n_i)$
tfc	$tf, idf, normalization$	$\frac{tf_i \cdot \log(N/n_i)}{\sum_i tf_i \cdot \log(N/n_i)}$

Table 4: Weighting schemes for visual-word feature [42]. Note that tf_i is number of times a visual word t_i appears in an image, N is the total number of images in the corpus, and n_i is the number of images having visual word t_i.

is originally used in speaker verification [32]. Because generally the number of features for only one is not enough to robustly learn the parameters of specialized GMM, in order to better build a specialized GMM for each video, there are two steps. The first step is to construct a global GMM on all available features. Secondly, specialized GMMs are derived from the global GMM in an Maximum a Posteriori way. Details of these two steps are presented as follows.

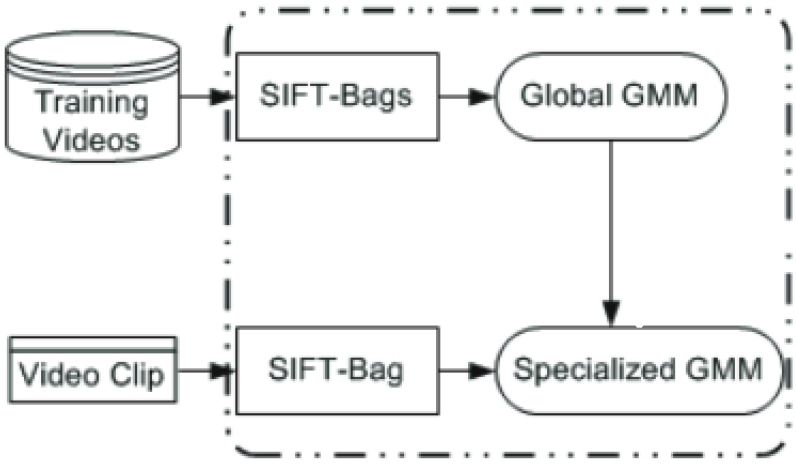

Figure 10: Illustration of building specialized GMMs for videos [45]. The first step is to build a global GMM based on SIFT features of training data. The second step is to derive a specialized GMM from the global GMM for each video clip.

Global GMM built from training data Once all SIFT features of training data are loaded into memory, K-means clustering algorithm is applied on this bag of SIFT features to initialize K centroids, where K is a predefined number determining the number of Gaussian components of the global GMM. Afterward, the standard Expectation-Maximization (EM) is applied to obtain the global GMM.

Suppose $X = \{x_1, \cdots, x_n\}$ are SIFT features extracted from training data, and $W = \{w_1, \cdots, w_K\}$ are the weights of respective Gaussian component. In E-step, the responsibilities of each SIFT feature to each Gaussian mixture component shall be computed. The below formula depicts the responsibility of x_i to Gaussian component k.

$$Pr(k|x_i) = \frac{w_k \cdot \mathcal{N}(x_i; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \mathcal{N}(x_i; \mu_j, \Sigma_j)} \tag{14}$$

$$n_k = \sum_{i=1}^{n} Pr(k|x_i) \tag{15}$$

In M-step, parameters of each Gaussian component are updated.

$$w_k = \frac{n_k}{n} \tag{16}$$

$$\mu_k = \sum_{j=1}^{n} \big(x_j \cdot \frac{Pr(k|x_j)}{n_k}\big) \tag{17}$$

$$\Sigma_k = \sum_{j=1}^{n} \big((x_j - \mu_k)(x_j - \mu_k)^T \cdot \frac{Pr(k|x_j)}{n_k}\big) \tag{18}$$

The above E-step and M-step are repeated until convergence, and the resulting distribution of X is modeled by Gaussian mixture models as

$$p(x_i; \Theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x_i; \mu_k, \Sigma_k) \tag{19}$$

where $\Theta = \{w_1, \mu_1, \Sigma_1, \cdots\}$, w_k, μ_k and Σ_k are the weight, mean and covariance matrix of the kth Gaussian component.

Specialized GMM by adaption The specialized GMM for video clip is built on the global GMM. Given $Z = \{z_1, \cdots, z_H\}$ as the SIFT features extracted from one video clip that is being modeled, a modified EM algorithm is used to calculate the specialized parameters for this video clip.

In E-step, the posterior probability is calculated as

$$Pr(k|z_i) = \frac{w_k \cdot \mathcal{N}(z_i; \mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \mathcal{N}(z_i; \mu_j, \Sigma_j)} \tag{20}$$

$$n_k = \sum_{i=1}^{H} Pr(k|z_i) \tag{21}$$

In M-step, mean vectors are updated as

$$E_k(Z) = \frac{1}{n_k} \sum_{i=1}^{H} Pr(k|z_i) z_i \tag{22}$$

$$\hat{\mu}_k = \alpha_k E_k(z) + (1 - \alpha_k)\mu_k \tag{23}$$

where $\alpha_k = {}^{n_k}/_{(n_k + r)}$, and r is adjusted empirically depending on H, the total number of SIFT features for each video clip.

The E-step and M-step are repeated until convergence. In the end, specialized GMM is modeled as $\hat{\theta} = \{\hat{\mu}_1, \cdots, \hat{\mu}_K\}$. It is worthy noting that all video clips are thus converted into equal size vectors if $\hat{\theta}$ is treated as a vector.

4.2 Distance Calculations

Now that videos are represented in either stack of histograms or specialized GMM, it is time to calculate video-to-video distance for these two representations. For bag of word model, EMD is incorporated into distance calculation. Given the property that EMD only needs the best matches between segments, the distance between videos with different frames could also be calculated. This idea is originally proposed by Xu and Chang [41], and it is now extended to Aligned Space-Time Pyramid Matching [11]. For specialized GMM representation, the author [45] propose a distance formula generated from Kullback-Leibler divergence based on the assumption that the covariance matrices are unchanged during the Maximum a Posteriori adaption process.

4.2.1 Aligned space-time pyramid matching

Let's say there are two videos: $P = \{p_1, \cdots, p_m\}$ and video $Q = \{q_1, \cdots, q_n\}$, where p_i and q_i are the respective histograms in P and Q. After attaching equal signatures to P and Q separately based on their number of histograms, P and Q are transformed into

$$P = \{(p_1, 1/m), ..., (p_m, 1/m)\}$$

$$Q = \{(q_1, 1/n), ..., (q_n, 1/n)\}$$

Given the above representation, the distance between P and Q are defined as

follows:

$$D_{PQ} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}} \tag{24}$$

where $[d_{ij}]$ is the ground distance matrix, d_{ij} calculates the distance between p_i and q_j, and $[f_{ij}]$ is the optimal flow that can be obtained by solving the below linear programming problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij} \\ \text{subjective to} \quad & f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n \\ & \sum_{j=1}^{n} f_{ij} \leq 1/m \quad 1 \leq i \leq m \\ & \sum_{i=1}^{m} f_{ij} \leq 1/n \quad 1 \leq j \leq n \\ & \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = 1 \end{aligned} \tag{25}$$

According to [11], the distance calculated through the above process refers to Level-0 distance, and the reason is because a video is treated as a whole rather than several subclips. To utilize the power of pyramid, each video clip is divided into 8^l nonoverlapped space-time volumes over multiple levels, $l = 1, \cdots, L-1$, where the volume size is set to be $1/2^l$ of the original video in width, height and temporal dimension. Figure 11 below demonstrates how video is divided into 8 sub videos at level one.

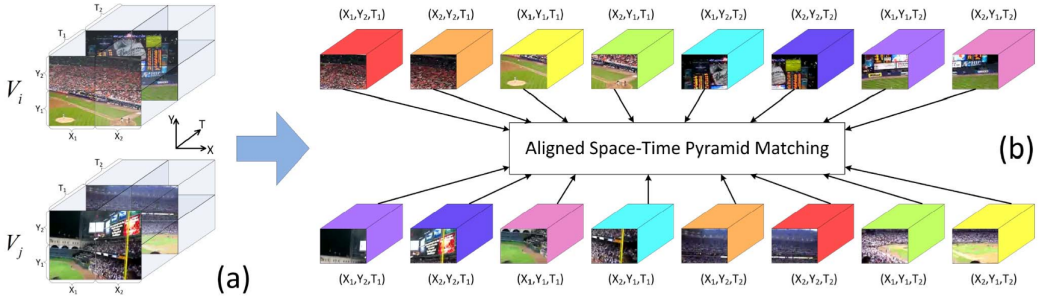

Figure 11: Illustration of Aligned Space-Time Pyramid Matching at level one [11]. (a) Each video clip is divided into 8 sub videos. (b) Matched sub videos are painted in the same color.

Compared with level zero distance, the calculation of higher level distances require one more matching stage due to the increasing volumes in videos. Let's say 2 videos P and Q are divided into 8^l nonoverlapped space-time volumes at level l, and they are represented as

$$P = (p_1, \cdots, p_R)$$

$$Q = (q_1, \cdots, q_R)$$

where p_i, q_j are divided space-time volumes and R equals to 8^l. There are two matching stages to calculate distance between P and Q at level l. The first one is to calculate pairwise distance matrix D, where D_{ij} is the EMD distance between p_i and q_j. As a result, the shape of D is R^2 because there are in total R^2 matches between space-time volumes of P and Q. The additional stage is to align these pairwise distances. After solving the below linear programming problem using integer-flow EMD, those space-time volumes in P and Q are explicitly aligned.

$$\begin{aligned} \hat{F}_{ij} = \arg\min_{F_{ij}} & \sum_{i=1}^{R}\sum_{j=1}^{R} F_{ij}D_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{R} F_{ij} = 1, \quad \forall j \\ & \sum_{j=1}^{R} F_{ij} = 1, \quad \forall i \end{aligned} \tag{26}$$

Finally, the aligned distance $D_l(P,Q)$ between P and Q at level l can be directly calculated as below

$$D_l(P,Q) = \frac{\sum_{i=1}^{R}\sum_{j=1}^{R} \hat{F}_{ij}D_{ij}}{\sum_{i=1}^{R}\sum_{j=1}^{R} F_{ij}} \tag{27}$$

4.2.2 Distances between specialized GMMs

Similar to previous section, let's say there are two video P and Q. They are represented as two specialized GMM shown as below.

$$P = (\mu_1^p, \cdots, \mu_K^p)$$

$$Q = (\mu_1^q, \cdots, \mu_K^q)$$

Given global GMM as $\Theta = \{w_1, \mu_1, \Sigma_1, \cdots\}$, the distance of P and Q is

$$d(P,Q) = \frac{1}{2}\sum_{k=1}^{K} w_k(\mu_k^p - \mu_k^q)^T\Sigma_k^{-1}(\mu_k^p - \mu_k^q) \tag{28}$$

Once video-to-video distances using above approaches are calculated, they will be transformed into Gaussian, Laplacian, ISD and ID kernels for recognition using SVM. Due to the poor performance of KNN, there is no experiment using KNN to recognize videos.

4.3 Other Approach: Concept Attributes

The approaches introduced above are under direct classification category, which model test videos in Bow or GMM and predict their labels directly. However, in

real word, many events in videos could be really complex. For example, in the behavior of "changing a vehicle tire", events like "person opening a car", "person using wrench" and "person holding something" may be involved. However, Bow and GMM representations fall to catch these low-level events. To be able to better understand complex events, researchers have proposed to use concept attributes to describe images and videos [22, 29, 28].

There are three stages to use concept attributes.

1. **Build concept detectors**
   Firstly, a concept space $\mathcal{C}^K$, where each dimension encodes the value of a semantic property, is defined. This space is spanned by K concepts $\{C_1, \cdots, C_K\}$. Next, K concept detectors are built based on training data. Each concept detector produces one score of respective concept to each input data sample.

2. **Represent data in concept attributes**
   Given the above K concept detectors, each data sample is input to these six detectors, and each detector produces one score. As a result, each data sample produces a vector $\{c_1, \cdots, c_k\}$, where c_i is the output score of ith concept detector.

3. **Recognize event classes using concept attribute representations**
   Now videos are transformed into vectors of concept attributes. Such representations are input into classification models like SVM for recognition.

There are many variations and improvements on the above three stages. For instance, detectors for images could also be used in videos, given that a video consists of multiple frames. If an image is represented as a K-dimensional vector, then a video is represented by a $W \times K$ matrix, where each row represents a vector of an image, and there are in total W images. For details, please refer to [22].

Last but not least, there are at least two advantages of using concept attributes. The first one is recognizing complex events, which has already been explained. Secondly, pre-trained detectors or even detectors shared by others could also be used to boost recognition performance. Given the fact that the cost of labelling training data is high, the second advantage provides lots of benefits by leveraging available resources.

5 Domain Adaptation

5.1 Overview

So far, all approaches introduced are based on an assumption tha both training data and testing data in recognition come from the same domain, namely, training data and testing data have the same or similar distribution. It works fine if there are enough lablled training data to build up a robust classifier. However, in real world, the most costing and time-consuming process is to label samples and build up training data. At the same time, because of some reasons, there are lots of labelled samples from other domains. For example, here is a task to recognize images from Instagram, and only few images are labelled. With such a small portion of labelled images, it is difficult to build a robust classifier. However, there are thousands of labelled images which could be retrieved through search engines like Google and Bing. Once certain keywords are input to the search engines, lots of relevant images will appear, and the keywords could be used as labels of these retrieved images. Due to the differences in distribution, classifiers built from other domains work poorly in the current domain. But is it possible to utilize these labelled samples from other domains to boost the performance of classifier in the current domain? To address this problem, many domain adaptation methods [9, 43, 10, 11], which utilizes training data from other domains, have been proposed for different proposes, and these methods have been proved to be effective to build robust classifiers with few labelled samples from the target domain.

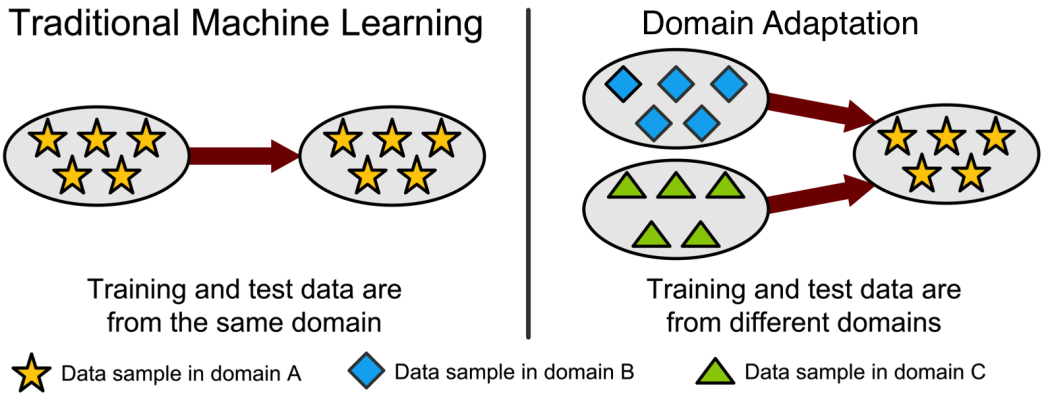

Figure 12: Comparison between traditional machine learning and domain adaptation. Data samples from other domains are also used as training data.

As shown in Figure 12, training data and testing data are supposed to come from the same domain in traditional machine learning. In contrast, domain adapta-

tion also utilizes data samples from other domains. Following the terminology from literatures, the target domain, from which the test samples are, is defined as $\mathcal{D}^T$. The other domain, which provides enough labelled data, is defined as auxiliary domain $\mathcal{D}^A$. Moreover, $\mathcal{D}^T = \mathcal{D}_l^T \cup \mathcal{D}_u^T$, where $\mathcal{D}_l^T$ represents limited labelled data, and $\mathcal{D}_u^T$ represents unlabelled data. Also, the auxiliary data set $\mathcal{D}^A = \{(x_i^A, y_i^A)\}_{i=1}^{n_A}$, which is a fully labelled data set. In the rest of this section, domain adaptation methods including Feature Replication [9], Adaptive SVM [43], Domain Transfer SVM [10] and Adaptive Multiple Kernel Learning [11] are introduced. Experimental results and analysis of these methods are presented in the section of experiments.

5.2 Feature Replication (FR)

Feature replication [9] uses augmenting features to perform SVM training. There are two different mapping functions to augment samples $\{x\}$ from different domains.

$$\Phi^T(\mathbf{x}) = (\mathbf{x}, \mathbf{x}, \mathbf{0}), \quad \Phi^A(\mathbf{x}) = (\mathbf{x}, \mathbf{0}, \mathbf{x}) \tag{29}$$

where Φ^T augments samples from $\mathcal{D}^T$, and Φ^A augments samples from $\mathcal{D}^A$. The kernelized version of the above transformation is pretty straightforward. Suppose the kernel used in SVM projects sample x into higher space through $\theta(x)$. Thus the kernel between x_i and x_j is: $K(x_i, x_j) = \theta(x_i)^T \cdot \theta(x_j)$. After the above augmenting process,

$$\begin{aligned} \Phi_\theta^T(x) &= (\theta(x), \theta(x), 0) \\ \Phi_\theta^A(x) &= (\theta(x), 0, \theta(x)) \end{aligned} \tag{30}$$

where $\Phi_\theta^T(x)$ represents x from $\mathcal{D}^T$ in higher dimensional space, and $\Phi_\theta^A(x)$ represents x from $\mathcal{D}^A$ in higher dimensional space. The expanded kernel is defined as $\hat{K}(x_i, x_j)$. If x_i and x_j come from the same domain,

$$\begin{aligned} \hat{K}(x_i, x_j) &= \theta(x_i)^T \cdot \theta(x_j) + \theta(x_i)^T \cdot \theta(x_j) \\ &= 2 \cdot \theta(x_i)^T \cdot \theta(x_j) \\ &= 2K(x_i, x_j) \end{aligned} \tag{31}$$

If x_i and x_j come from different domains,

$$\begin{aligned} \hat{K}(x_i, x_j) &= \theta(x_i)^T \cdot \theta(x_j) \\ &= K(x_i, x_j) \end{aligned} \tag{32}$$

To summarize,

$$\hat{K}(x_i, x_j) = \begin{cases} 2K(x_i, x_j) & \text{if } x_i \text{ and } x_j \text{ come from the same domain} \\ K(x_i, x_j) & \text{otherwise} \end{cases}$$

Based on equations (31) and (32), considering the kernel as a measure of "similarity", samples from the same domain are twice as similar as those come from different domains. In order words, for a test sample, those training samples from the same domain are going to be more influential. Furthermore, the implementation is rather easy since the only operation needed is to twice the kernels of samples from the same domain.

5.3 Adaptive Support Vector Machine (A-SVM)

Adaptive SVM [43] incorporates multiple auxiliary classifiers $f_1^A(x), \cdots, f_M^A(x)$ built from $\mathcal{D}^A$ into the standard structure of SVM. The decision function of adapted classifier is defined as

$$f(x) = \sum_{k=1}^{M} t_k f_k^A(x) + \Delta f(x) \tag{33}$$

where $\Delta f(x) = w^T \phi(x)$ is called as a *permutation function* which is learned from labelled data in the target domain ($\mathcal{D}_l^T$) only, $t_k \in (0, 1)$ is the weight of each auxiliary classifier which sums to one. Because of the change in decision function, the objective function of SVM is changed to be

$$\begin{aligned} &\text{minimize} \quad J(w) = \tfrac{1}{2}\|w\|^2 + C\textstyle\sum_{i=1}^{N} \xi_i \\ &\text{subject to} \quad \xi_i \geq 0, \quad y_i \textstyle\sum_{k=1}^{M} t_k f_k^A(x_i) + y_i w^T \phi(x_i) \geq 1 - \xi_i \quad \forall i \end{aligned} \tag{34}$$

From the objective function listed above, it is not hard to distinguish that Adaptive SVM aims to find a hyperplane which is not only close to hyperplanes built from $\mathcal{D}^A$ but also separates labelled samples in $\mathcal{D}^T$ well. Due the change of objective function, the Lagrange dual form is changed to be

$$W(\alpha) = \sum_i (1 - \lambda_i)\alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \tag{35}$$

where $\lambda_i = y_i \sum_{k=1}^{M} t_k f_k^A(x_i)$. Once the set $\hat{\alpha}$ which maximizes $W(\alpha)$ is found, the decision function $f(x)$ is expressed as

$$f(x) = \sum_{k=1}^{M} t_k f_k^A(x) + \sum_i \hat{\alpha}_i y_i K(x_i, x) \tag{36}$$

5.4 Domain Transfer Support Vector Machine (DTSVM)

It has been emphasized that the distributions of $\mathcal{D}^T$ and $\mathcal{D}^A$ are different. Therefore, when these two sets of data are projected into higher dimensional space, their distributions are still distant to each other. If there is a way to reduce the mismatch between $\mathcal{D}^T$ and $\mathcal{D}^A$, then those lablled samples from $\mathcal{D}^A$ could be treated as samples in the target domain to some extent, and of course this is going to improve the performance of recognition because more useful training samples are input to build a robust classifier. Given the fact that fusing multiple kernels helps to increase the performance, Duan et al. [10] proposed Domain Transfer SVM (DTSVM) to reduce the mismatch between $\mathcal{D}^T$ and $\mathcal{D}^A$ through calculating the optimal weight of each kernel. Also, the respective classifier is learnt simultaneously.

The mismatch measured by Maximum Mean Discrepancy (MMD) [4] is defined as follows

$$DIST_k(\mathcal{D}^A, \mathcal{D}^T) = \| \frac{1}{n_A} \sum_{i=1}^{n_A} \varphi(x_i^A) - \frac{1}{n_T} \sum_{i=1}^{n_T} \varphi(x_i^T) \| \tag{37}$$

where x_i^A and x_i^T are samples from the auxiliary and target domains, respectively, and $\varphi(x)$ is a mapping function which projects x into higher dimensional space. Given this mapping function, the kernel between x_i and x_j is $k(x_i, x_j) = \varphi(x_i)^T \cdot \varphi(x_j)$. To simplify equation (37), a column vector $\mathbf{s}$, in which the first n_A entries are $\frac{1}{n_A}$ while the remaining n_T entries are $-\frac{1}{n_T}$, is used. Now the square of MMD in equation (37) is simplified as

$$DIST_k^2(\mathcal{D}^A, \mathcal{D}^T) = \mathrm{tr}(\mathbf{KS}) \tag{38}$$

where $\mathbf{S} = \mathbf{ss^T}$, and $\mathbf{K} = \begin{bmatrix} K^{A,A} & K^{A,T} \\ K^{T,A} & K^{T,T} \end{bmatrix}$, and $K^{A,A}$, $K^{T,T}$ and $K^{A,T}$ are kernel matrices defined for auxiliary domain, target domain and the cross domain for the auxiliary domain to target domain.

According to traditional multiple kernel learning assumption [16], the final kernel

k is a result of linear combination of several base kernels $k_1, \cdots, k_M$.

$$k = \sum_{m=1}^{M} d_m k_m \tag{39}$$

where d_m is the coefficient of k_m with constraints $d_m \geq 0$ and $\sum_{i=1}^{M} d_m = 1$. Based on final kernel k, the decision function is thus defined as

$$f^T(x) = \sum_{m=1}^{M} d_m w_m^T \varphi_m(x) + b \tag{40}$$

As a result, once the optimal coefficients $\mathbf{d} = [d_1, \cdots, d_M]^T$ are found through the process of reducing mismatches between $\mathcal{D}^T$ and $\mathcal{D}^A$, this decision function will be easily defined. After incorporating equation (39) into (38), the square of MMD is defined as a function of $\mathbf{d}$.

$$DIST_k^2(\mathcal{D}^A, \mathcal{D}^T) = \Omega(\mathbf{d}) = \mathbf{h^T d} \tag{41}$$

where $\mathbf{h} = [tr(\mathbf{K_1 S}), \cdots, tr(\mathbf{K_M S})]^T$, and $\mathbf{K_m} = [\varphi(x)^T \varphi(x)]$ is the mth base kernel matrix defined on samples from both auxiliary and target domains. After all these settings, the optimization problem of DTSVM is defined as

$$\text{minimize} \quad G(\mathbf{d}) = \frac{1}{2}\Omega^2(\mathbf{d}) + \theta J(\mathbf{d}) \tag{42}$$

where θ is is a tradeoff parameter which balances the mismatch of distributions and the structural risk function of SVM, and

$$J(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j (\sum_{m=1}^{M} d_m \ \varphi_m(x_i)^T \varphi_m(x_j)) \tag{43}$$

To solve the optimization problem, the author [10] employed the reduced gradient descent procedure to iteratively update coefficients $\mathbf{d}$ and dual variable $\boldsymbol{\alpha}$. There are two steps:

1. **Update the dual variable $\boldsymbol{\alpha}$**
   Given the current value $\mathbf{d}$, the optimization problem in equation (43) are solved by LIBSVM [7], and the dual variable $\boldsymbol{\alpha}$ is updated.

2. **Update the linear combination coefficients d**

At $t+1$ iteration, d_{t+1} is updated as

$$\mathbf{d}_{t+1} = (1-\eta_t)\mathbf{d}_t + \eta_t \mathbf{d}_t^{new} \tag{44}$$

where $\mathbf{d}_t^{new} = \theta(\mathbf{h}\mathbf{h}^{\mathbf{T}} + \varepsilon\mathbf{I_M})^{-1}\mathbf{q}$, and $\varepsilon\mathbf{I_M}$ is added to avoid numerical instability with $\varepsilon = 10^{-5}$, and $\mathbf{q} = [\frac{1}{2}(\boldsymbol{\alpha}_t \diamond \mathbf{y})^T \mathbf{K}_1 (\boldsymbol{\alpha}_t \diamond \mathbf{y}), \cdots, \frac{1}{2}(\boldsymbol{\alpha}_t \diamond \mathbf{y})^T \mathbf{K}_M (\boldsymbol{\alpha}_t \diamond \mathbf{y})]$, and η_t is the learning rate. Please note the symbol $\diamond$ represents the element-wise product of two vectors. For formal mathematical induction, please refer to [10, 11] for details, and a formal and concise algorithm description is presented in [11].

The above two steps are repeated until convergence or the maximum number of iteration allowed is reached. At the end of this algorithm, the optimal $\mathbf{d}$ is learned as well as the optimal SVM model.

5.5 Adaptive Multiple Kernel Learning (A-MKL)

Inspired by adaptive SVM (A-SVM) [43] and domain transfer SVM (DTSVM) [10], the authors of DTSVM creatively combined these two approaches and proposed adaptive multiple kernel learning (A-MKL) [11]. A-MKL takes the advantage of pre-learnt classifiers built either from $\mathcal{D}^A$ or $\mathcal{D}^A \cup \mathcal{D}^T$ and defines the decision function as

$$f^T(x) = \sum_{p=1}^{P} \beta_p f_p(x) + \sum_{m=1}^{M} d_m w_m^T \varphi_m(x) + b \tag{45}$$

Compared with equation (40), the added item $\sum_{p=1}^{P} \beta_p f_p(x)$, where $f_1(x), \cdots, f_P(x)$ are pre-learnt classifiers with β being the weight of each classifier, represents the influence from pre-learnt classifiers. Because of the influence coming from pre-learnt classifiers, compared with DTSVM, the difference is that equation (43) becomes

$$J(\mathbf{d}) = \max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \Big(\sum_{m=1}^{M} d_m \breve{\varphi}_m(x_i)^T \breve{\varphi}_m(x_j)\Big) \tag{46}$$

where $\breve{\varphi}_m(x_i)^T \breve{\varphi}_m(x_j) = \varphi_m(x_i)^T \varphi_m(x_j) + \frac{1}{\lambda} \boldsymbol{f}(x_i)^T \boldsymbol{f}(x_j)$, and $\boldsymbol{f}(x)$ is a vector of predictions on x from pre-learnt classifiers $f_1(x), \cdots, f_P(x)$, and λ is a regularization parameter in the objective function of SVM. Since the only change is the kernel matrix, once the kernel matrix is fused by scores of pre-learnt classifiers, the algorithm used in DTSVM could be directly employed to calculate the

optimal coefficients $\mathbf{d}$ and dual variables $\boldsymbol{\alpha}$. With the help from scores output by pre-learnt classifiers, A-MKL not only reduces the mismatch of distributions between $\mathcal{D}^A$ and $\mathcal{D}^T$ but also finds a hyperplane which is close to the optimal hyperplane of $\mathcal{D}^A$ in "new" distribution.

6 Experiments

6.1 Image Recognition

6.1.1 Data set description

The image data set, in which there are six classes (shooting, playing guitar, running, phoning, riding bike and riding horse) as shown in Figure 13, comes from [20]. For each class, there are 60 images with different backgrounds, different persons and different viewpoints.

Figure 13: Samples from the image data set used in experiments [20]

6.1.2 Performance of spatial pyramid matching

To examine the performance of spatial pyramid matching, a vocabulary with 300 was built from all SIFT features extracted from the total 360 images using Mini-Batch K-Means cluster algorithm. Next, three kinds of pyramids were built: one-level, two-level and three level. In the process of building pyramids, histograms at different levels were concatenated together to form a long vector with respective weights. As a result, one-level pyramid was a 300-dimensional vector, two-level pyramid was a 1500-dimensional vector and three-level pyramid was a 6300-dimensional vector. As for classification, SVM with 5 different kernels and KNN were employed based on different pyramid representations. In this experiment, 40 images each class were randomly chosen to act as training data while the left 20 images each class were used as testing data set. Also, the recognition accuracy is defined as

$$\text{recognition accuracy} = \frac{\text{correct predictions}}{\text{number of testing samples}} \tag{47}$$

where correct predictions represent the number of testing samples which are correctly predicted. The recognition results using the above settings are reported in

Table 5 and 6.

SVM	One-level	Two-level	Three-level
Linear	74.17	80	**87.5**
Poly	70	62.5	31.67
RBF	37.5	83.33	74.17
Sigmoid	16.67	16.67	16.67
Histogram Intersection	79.17	82.5	**85**

Table 5: Recognition accuracies (percent) of different spatial pyramids using SVM with different kernels

	One-level	Two-level	Three-level
KNN	50.83	45	48.33

Table 6: Recognition accuracies (percent) of different spatial pyramids using KNN

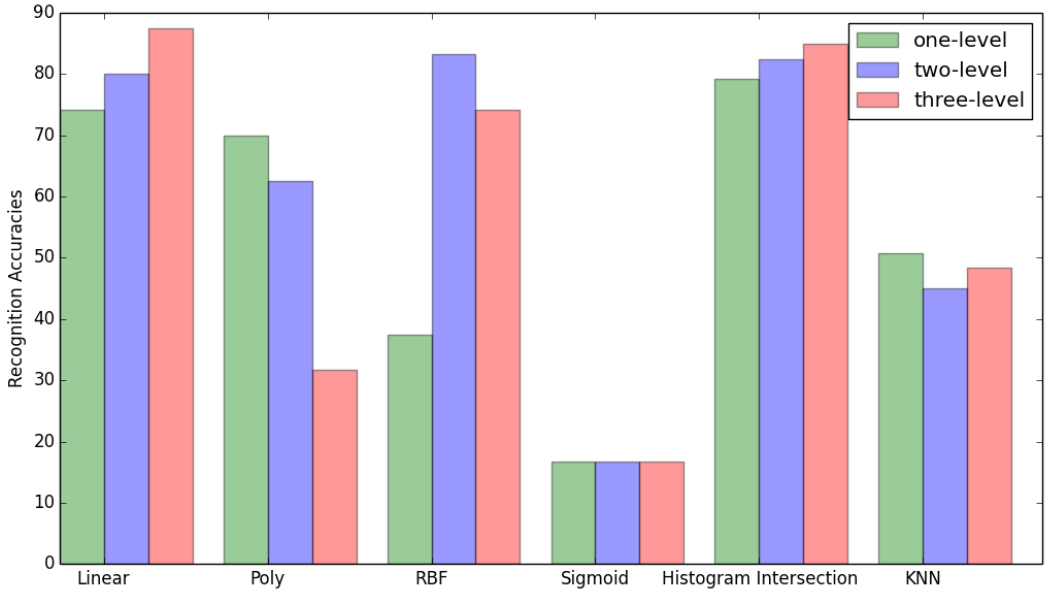

Figure 14: Chart representation of combined Table 5 and Table 6

Table 5 above reports the recognition accuracies using SVM with different kernels for different pyramids. The kernel types which projects samples into higher dimensional space like "Poly" and "RBF" abd "Sigmoid" performed much worse than "Linear" and "Histogram Intersection". Also, it is easy to see that the recognition accuracies increased by fusing information from multiple levels. For

"Linear" and "Histogram Intersection", their performances were improved significantly from one-level pyramid to three-level pyramid, and "Linear" achieved the best result 87.5% while "Histogram Intersection" achieved the second best result 85% at three-level pyramid. Table 6 depicts the performance of KNN when the number of neighbors was set to 5. Compared with reasonable result of SVM, KNN did not perform quite well. The best result achieved at one-level 50.83% is much smaller than 74.17% which was achieved by SVM with "Linear" kernel. Such result makes sense because it is generally thought that KNN performs worse than SVM since KNN is a lazy learner which does not consider the global information.

6.1.3 Performance of earth mover's distance

EMD was also experimented to examine its effectiveness in recognition. Following the method introduced before, each image was divided into 4 pieces equally, and EMD attempted to align these 4 pieces. In this experiment, the distance matrix calculated with EMD was referred as aligned distance while the distance matrix without EMD was referred as unaligned distance. Once distances were calculated, they were transformed into four different gram matrices through four different kernel functions (RBF, LAP, ID and ISD). Finally, these kernel matrices were input into SVM for classification. Unlike previous division of training and testing samples, this experiment randomly selected 50% images as training data and the other 50% as testing data. The result using the above settings is recorded in Table 7.

	RBF	**LAP**	**ID**	**ISD**
Aligned Distance	79.44	76.11	73.89	75.56
Unaligned Distance	78.89	75.56	73.33	75

Table 7: Comparison of recognition accuracy (percent) between aligned distance and unaligned distance

For all four kernel types, aligned distance performed better than unaligned distance. Though slightly, it shows the robustness of EMD which gives a more accurate image-to-image distance. Also, it is worth noting that these four kernels output different results. So it is important to select kernel function in order to build a robust classifier. Given the fact that there are different kernels, is it possible to combine them? The answer is yes, and it refers to multiple kernel learning, which will be experimented in the following subsection on video recognition.

6.2 Video Recognition

6.2.1 Data set description

The data set used in this experiment was retrieved from Visual Computing Group of Nanyang Technological University, and this data set had already been used for comprehensive experiments in [11]. There are two different domains [24]: one comes from the Kodak Consumer Video Benchmark Data set while the other comes from Youtube. Also, there are two types of features: one is SIFT while the other one is space-time feature. According to the accuracies of different features reported in [11], SIFT performed much better than space-time features. As a result, SIFT was chosen to conduct experiments. There are six different classes, and the numbers of videos in each class from different domains are summarized in the below table.

	Wedding	**Sports**	**Show**	**Picnic**	**Parade**	**Birthday**	**Total**
Kodak	27	75	57	6	14	16	195
Youtube	91	260	200	85	119	151	906

Table 8: Number of videos in each class from Kodak and Youtube

6.2.2 Performance of aligned space-time pyramid matching

To examine Aligned Space-Time Pyramid Matching, a visual vocabulary was firstly needed in order to convert videos into stacks of histograms. The visual vocabulary with 2500 words was built on SIFT features sampled from features of Kodak data set. Due to a lack of physical memory, the size of training data was only 167837×128, where 167837 was the number of SIFT features. Although the training size was relatively small, fortunately, it still produced comparable results compared with that reported in [11]. The next step was to convert all Kodak videos into either 1 stack of histograms (level 0) or 8 stacks of histograms (level 1) based on the visual vocabulary built before. Once BoW models for all Kodak videos were built, EMD was employed to calculated video-to-video distances among all these videos. In this experiment, the C program wrote by Rubner, who is the proposer of Earth Mover's distance [33], was used. To integrate this C program with the main codes written in Python, an interface was added to enable communications between C and Python codes. Please also note that the time taken by calculating level one distances is around 65 times of that taken by calculating level 0 distance. When calculating level one distance, the number of

pairwise distances needed is $8 \times 8 = 64$. If the aligned process is also considered, the number of times to invoke EMD becomes 65. However, for level 0 distance, only one time is needed to call EMD when calculating the distance between two videos. Similar to the settings in image recognition, distances were then transformed into gram matrices using Gaussian, Laplacian, ISD and ID. The final step was to employ SVM to perform recognition based on these gram matrices. Due to the poor performance of KNN, KNN was not tested here. Furthermore, a new mode fusing SVM scores from Gaussian, Laplacian, ISD and ID was tried, and this fusing process is defined as

$$f^{Fuse} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + \exp(-f_i)} \tag{48}$$

where N is the number of classifiers, and f_i is the score output by ith classifier. In this case, $N = 4$, and the four SVM classifiers were built from Gaussain, Laplacian, ISD and ID kernel matrices.

Based on scores output by different settings, Average Precision (AP), which is the most widely used performance metric of video event recognition, was calculated. Denote R as the number of true relevant videos in the dataset. At any index j, let R_j be the number of true relevant videos in the top j list. Average Precision is defined as

$$AP = \frac{1}{R} \sum_{j} \frac{R_j}{j} \times I_j \tag{49}$$

where $I_j = 1$ if and only if the jth video is true relevant, otherwise $I_j = 0$.

	Gaussian	**Laplacian**	**ISD**	**ID**	**Fused scores**
Level 0	44.38 ± 2.13	44.90 ± 2.73	44.01 ± 2.13	45.36 ± 3.13	44.33 ± 2.61
Level 1 (Unaligned)	43.08 ± 3.14	43.85 ± 3.84	43.22 ± 3.11	43.85 ± 3.56	43.55 ± 3.46
Level 1 (Aligned)	43.61 ± 2.97	43.40 ± 3.18	43.46 ± 2.97	43.22 ± 3.11	44.08 ± 3.25

Table 9: Means and standard deviations (percent) of MAPs over six events at different levels using SVM with different kernels.

In the experiment shown in Table 9, each time 3 videos from each class were randomly selected as training data while the rest of Kodak videos were used as testing videos. As a result, 18 videos were used to train models while the rest 177 videos acted as testing videos. Furthermore, SVM was trained in one-against-all manner, which trained in total 6 classifiers for 6 events. As a result, each classifier produced one AP, and the mean of these 6 APs (MAP) was recorded.

This experiment was repeated for five times, and the mean and standard deviation of Mean Average Precision (MAP) under each setting was recorded in Table 9. Given the fact that only around 9.23% ($^{18}/_{195}$) videos were used as training data, the performance reported in Table 9 was quite good. There are three observations based on results in Table 9.

1. **In this case, Level 0 performs better than both unaligned Level 1 and aligned Lavel 1 for all types of kernel**
   The most possible reason is due to a lack of training data. In latter experiments, when all youtube videos and randomly selected 18 Kodak videos were used for training, the performance ranking was: Level 1 (Aligned) > Level 1 (Unaligned) > Level 0.

2. **Level 1 (Aligned) performed better than Level 1 (Unaligned) in most cases**
   Thus, the one additional alignment makes worthwhile contribution to a better performance though it requires one more call of EMD.

3. **The approach of fusing scores provides a guarantee on performance**
   For all three cases, the means of MAP using fused scores were always not the worst, and it was even the best for aligned Level 1 distance. Therefore, it provides a good insight that fusing scores of multiple classifiers is a good alternative if the standalone performances of those classifiers are unclear.

	Gaussian	**Laplacian**	**ISD**	**ID**	**Fused scores**
bxx	40.20 ± 2.57	38.35 ± 2.31	39.93 ± 2.58	38.23 ± 2.08	39.34 ± 2.55
txx	44.28 ± 2.14	44.90 ± 2.73	44.01 ± 2.13	45.36 ± 3.13	44.33 ± 2.61
txc	42.15 ± 4.73	45.01 ± 3.45	43.47 ± 4.56	45.38 ± 3.20	44.11 ± 3.90
tfx	43.76 ± 2.99	44.14 ± 3.36	43.61 ± 3.03	44.05 ± 3.51	44.18 ± 3.22
tfc	43.71 ± 1.37	$\mathbf{46.02 \pm 1.84}$	$\mathbf{44.93 \pm 1.64}$	$\mathbf{46.21 \pm 1.83}$	$\mathbf{45.28 \pm 1.62}$
Easy soft	43.54 ± 2.12	44.77 ± 2.41	43.52 ± 2.08	45.24 ± 2.47	44.79 ± 2.55
Gaussian soft	$\mathbf{44.77 \pm 2.80}$	45.23 ± 2.76	44.90 ± 3.01	45.23 ± 2.87	$\mathbf{45.20 \pm 3.04}$

Table 10: Means and standard deviations (percent) of MAPs over six events using different mechanisms to build histograms

Experiments on building better BoW using different weighting schemes and soft assignment were also tried. Following the same setting of experiments in Table 9, the results of various approaches were recorded in Table 10. "Easy soft" represents the approach that the top T neighbors are retrieved for each feature when

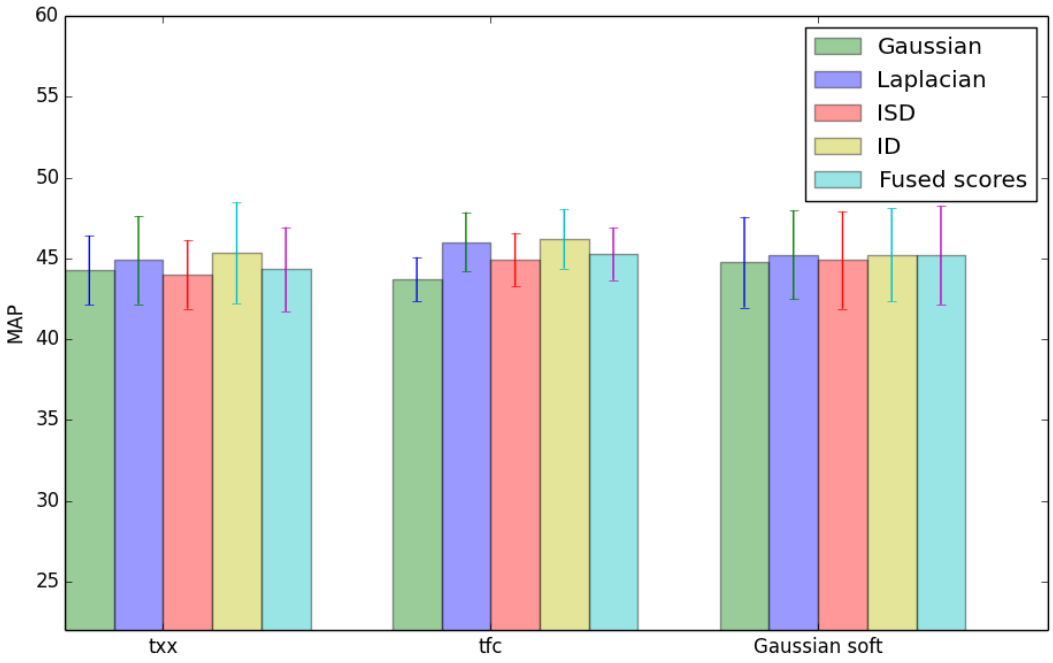

Figure 15: Chart representation of MAPs using txx, tfc, Gaussian soft

building histograms. In this experiment, $T = 4$. "Gaussian soft" represents the approach which uses Gaussian Mixture Models to build histograms. For other approaches like "txc", please refer to Table 4 in the section of representations of videos. Based on results shown in Table 10, it is not easy to distinguish that "tfc" and "Gaussian soft assignment" performs relatively better than the rest approaches. Thus, histograms built through soft assignment and weighting schemes which consider global frequencies of visual words poss more discriminable power.

Training videos	Testing videos	Original videos	Compressed videos
60	846	38.9 ± 2.9	38.6 ± 2.8
120	786	45.7 ± 2.2	44.5 ± 1.6
180	726	49.5 ± 1.8	48.3 ± 1.9
240	666	52.0 ± 2.1	50.6 ± 2.1

Table 11: Means and standard deviations (percent) of MAPs over six events using distance matrix calculated on original Youtube videos and distance matrix calculated on Youtube compressed videos under different distributions of training and testing videos.

The approach of identifying key frames in a cluster of images introduced in the section of Image Recognition was employed to compress Youtube videos. Originally, the total size of Youtube histograms in form of Python serialization file was 4.42 GB. After compression, the size was reduced to 3.17 GB, which means around

28.41% frames were discarded. Then the level 0 distance matrix of compressed Youtube videos using Aligned Space-Time Pyramid Matching was calculated. This experiment followed previous experimental setting and still employed MAP as performance metric. Furthermore, the distance matrix of original Youtube videos were used as a baseline. Table 11 depicts means and standard deviations of MAPs for these two distance matrices using different number of training videos. There are two observations based on Table 11.

1. **The performance of compressed videos is slightly worse than original videos**
   Though compressed videos contain less redundant frames, the performance is still worse than original videos. The most possible explanation is that EMD always finds the best matches among frames. Thus, those redundant frames are actually ignored during the calculation of distances. As a result, the compressing step may not be so useful even though it accelerates the calculation of EMD distance because of smaller data size. This observation also further proves the effectiveness of EMD.

2. **With more training samples, the performance becomes better**
   This is actually common sense and easy to understand. With more training samples, the found boundary in the hyperplane is more likely to be near to the optimal boundary. Based on this observation, in order to build robust classifiers, one approach is to employ more training samples. However, the problem is there is normally no enough labelled samples, and it costs a lot to label new samples. This is also the main reason why researchers are exploring domain adaptations, in which labelled samples in other domains are leveraged.

6.2.3 Performance of specialized GMM

	Gaussian	**Laplacian**	**ISD**	**ID**	**Fused scores**
spherical 128	24.70 ± 1.41	43.04 ± 1.61	26.92 ± 1.00	$\mathbf{43.64 \pm 0.96}$	32.91 ± 2.20
spherical 64	23.99 ± 1.40	42.35 ± 1.64	25.62 ± 1.11	43.42 ± 1.18	29.01 ± 1.10
full 128	25.69 ± 7.57	21.39 ± 7.32	26.49 ± 8.38	21.93 ± 7.75	21.79 ± 7.29
full 64	25.23 ± 0.94	29.69 ± 1.81	25.68 ± 1.34	30.74 ± 1.67	26.74 ± 1.63

Table 12: Means and standard deviations (percent) of MAPs over six events using different GMMs

Experiments using another representations of video clips are presented here. There are in total four different settings controlled by the type of covariance

and the number of dimension in SIFT feature. Two types of covariances were adopted in these experiments: spherical and full. While spherical covariance assumes that the variations on all dimensions are the same, full covariance does not have such assumption. An example to demonstrate the differences of these two types of covariances is Figure 16, where spherical covariance builds Gaussian distributions in circle shape, and full covariance builds in ellipse. Rather than the original 128 dimensional SIFT, SIFT reduced to 64 dimensional feature through PCA was also tried in experiments.

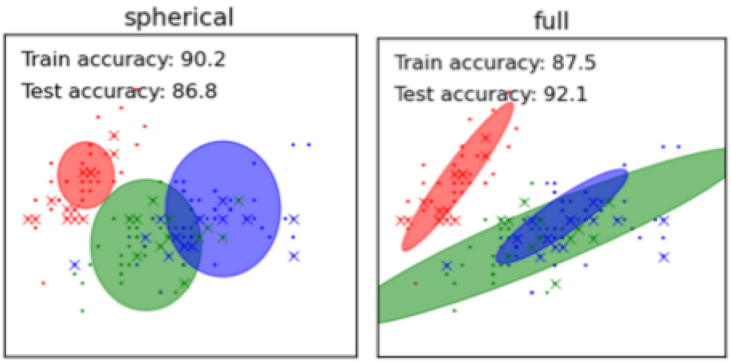

Figure 16: Gaussian Mixtures Models built by different covariance types [31].

Based on results in Table 12, there are two observations.

1. **Spherical covariance performed better than full covariance**
   Except for Gaussian kernel, the performances of spherical covariance with 128 dimensional SIFT features were better than those of full covariance. The possible reason could be that the number of features was not enough to build good enough specialized GMM for most video clips when full covariance was adopted.

2. **Spherical covariance using ID kernel performed the best**
   Overall speaking, GMM representations of videos did not perform well compared with BoW representations. However, the best MAP achieved by spherical covariance using 128 dimensional feature reached 43.64%, which is slightly worse than that achieved by level 0 distance using aligned space-time pyramid matching. This gives an insight that an appropriate kernel function helps a lot to improve the performance.

	Recognition accuracy
Kodak → Kodak	38.5 ± 12.7
Youtube → Kodak	30.0 ± 6.9
Baseline	41.6 ± 11.5

Table 13: Means and standard deviations (percent) of recognition accuracies using concept attributes

6.2.4 Performance of concept attributes

Concept attribute was also experimented. In this experiment, 2-layer SVM classification structure was implemented, where the first layer built up 6 concept detectors, and the second layer trained and tested videos using 6-dimensional vectors, in which each dimension represented the respective semantic concept like "birthday" and "show". The results were presented in Table 13. "Kodak → Kodak" represents the case where training samples in Kodak videos were used to train concept detectors, whereas "Youtbe → Kodak" represents the case where all Youtube videos were used to train concept detectors. The second stage SVM was implemented in one-against-one manner with Gaussian kernel. As for baseline, it was implemented using level 0 distance and only Kodak videos as training data through one-against-one SVM.

Based on results recorded in Table 13, there are two observations.

1. **"Youtbe → Kodak" performed much worse than "Kodak → Kodak"**
   This result makes sense since Youtbe videos and Kodak videos come from two different domains. Due to the differences in distributions, the concept detectors built through Youtube videos may not be powerful enough to capture adequate discriminate power.

2. **"Kodak → Kodak" was slightly worse than baseline**
   This observation demonstrates the promising discriminable power of concept attribute. Though the representation of videos is reduced into 6-dimensional vectors, its performance is still comparable with that achieved by baseline. Actually, concept attribute could be implemented to perform better by adding more concepts. However, there was no enough training data.

6.2.5 Performance of domain adaptation methods

This subsection introduces the experiments on domain adaptation methods. There are in total four domain adaptations: Feature Replication (FR), Adaptive SVM (A-SVM), Domain Transfer SVM (DTSVM) and Adaptive Multiple Kernel Learning (A-MKL). The other three methods SVM_T, SVM_AT and MKL are used as baseline, where SVM_T only relies on $\mathcal{D}_l^T$, and SVM_AT relies on training samples from both $\mathcal{D}^T$ and $\mathcal{D}^A$, and MKL uses the average of all kernel matrices. Therefore, except SVM_T, all other 6 methods needed all Youtube videos as training data in $\mathcal{D}^A$. In Table 14, various distance matrices are listed. It is worth mentioning the level 1 distances in categories MAP(2) and MAP(3). It has already been introduced that the calculation of level 1 distance is 64 times slower than that of calculating level 0 distance. It was estimated to cost around 20 days to compute level 1 distances between all 1101 videos on single machine, and this fact was not acceptable. In order to overcome this difficulty, multi-process was employed, and this method successfully decreased the cost to around 6 days.

Once a distance matrix was given, 20 kernel matrices were built, where Gaussian, Laplacian, ISD and ID were employed by setting kernel parameter as $\gamma = 2^l\gamma_0$, and $l \in (-3, -2, \cdots, 1)$. Such setting was made to align with that stated in [11]. One special case was MAP(8) in which 2 different distance matrices were input. As a result, there were in total 40 base kernels for MAP(8) during the process of all these methods. As for the division of training samples and testing samples, each time the experiment randomly selected 3 videos from Kodak set for each class as $\mathcal{D}_l^T$ while the left videos were treated as $\mathcal{D}_u^T$. All Youtube videos were used as full labelled set in $\mathcal{D}^A$ if needed. The results of repeating this experiment for five times were recorded in Table 15 as shown below.

Table 15 gives the means and standard deviations of MAPs over six events for all methods, and there are 4 observations.

1. **In all cases, SVM_AT performed better than SVM_T**
   This result is achieved as expected, and it demonstrates the limitation of classifiers built from limited training samples in $\mathcal{D}^T$. If there are more labelled samples from $\mathcal{D}^A$ used in training, the resulting classifiers are generally more robust.

2. **Among almost all 7 methods from SVM_T to A-MKL, the distance matrix calculating from histograms built by Gaussian soft**

Setting Name	Content
MAP(1)	Level 0 distance in aligned space-time pyramid matching
MAP(2)	unaligned Level 1 distance in aligned space-time pyramid matching
MAP(3)	aligned Level 1 distance in aligned space-time pyramid matching
MAP(4)	distances calculated by histograms built in "tfc" weighting scheme
MAP(5)	distances calculated by histograms built by straightforward soft assignment
MAP(6)	distances calculated by histograms built by GMM soft assignment
MAP(7)	distances calculated by specialized GMMs built on 128 dimensional SIFT features with spherical covariance
MAP(8)	two set of distances: aligned Level 1 distances and distances from histograms built through GMM soft assignment

Table 14: Experimental settings

	SVM_T	SVM_AT	FR	A-SVM	MKL	DTSVM	A-MKL
MAP(1)	44.33 ± 2.61	52.21 ± 2.54	52.33 ± 2.20	47.03 ± 3.26	50.82 ± 2.16	47.14 ± 3.26	54.29 ± 2.21
MAP(2)	43.55 ± 3.46	55.37 ± 2.26	55.95 ± 3.79	45.86 ± 4.39	55.08 ± 6.17	50.97 ± 1.38	54.26 ± 3.46
MAP(3)	44.08 ± 3.25	57.56 ± 3.02	53.91 ± 1.48	45.42 ± 3.62	53.81 ± 3.86	53.32 ± 2.56	$\mathbf{57.45 \pm 1.64}$
MAP(4)	45.27 ± 1.63	51.83 ± 2.27	52.55 ± 2.00	45.94 ± 1.70	51.70 ± 2.35	52.31 ± 2.56	53.05 ± 2.21
MAP(5)	44.79 ± 2.55	47.80 ± 1.67	51.89 ± 1.99	47.41 ± 3.13	47.16 ± 1.77	45.05 ± 4.07	51.08 ± 2.87
MAP(6)	45.20 ± 3.04	56.90 ± 2.79	54.03 ± 4.02	46.62 ± 3.14	55.47 ± 4.11	53.41 ± 3.29	$\mathbf{59.16 \pm 3.38}$
MAP(7)	32.91 ± 2.20	33.15 ± 1.78	41.78 ± 3.98	37.07 ± 3.52	33.64 ± 0.74	$\mathbf{46.61 \pm 2.41}$	35.88 ± 1.98
MAP(8)	44.69 ± 2.84	60.21 ± 1.94	55.29 ± 3.00	46.28 ± 4.23	58.36 ± 3.75	57.01 ± 2.45	$\mathbf{61.40 \pm 1.91}$

Table 15: Means and standard deviations (percent) of MAPs over six events for all methods

assignment(MAP(6)) performed better than other 6 distance matrices from MAP(1) to MAP(5) and MAP(7)

This statement is made without considering MAP(8) which used 2 kinds of distance matrices. This result strongly shows the robustness of histograms built through Gaussian soft assignment. During the process of building histograms through Gaussian soft assignment, more discriminable power is retained compared with other approaches.

3. **A-MKL performed the best**

   MAP(8) achieved the best using A-MKL, where aligned Level one distance and distance from Gaussian soft assignment were used. A-MKL utilizes information from pre-learnt classifiers and reduces the mismatch between $\mathcal{D}^T$ and $\mathcal{D}^A$ by finding the optimal coefficients of multiple base kernels. Though there are at least 20% noisy labels in Youtube videos as reported in [11], A-MKL still learned a robust classifier and achieved the best. The best MAP 61.40% is better than the results reported in [11], and it demonstrates the effectiveness of selecting base kernels smartly.

4. **DTSVM performed amazingly in MAP(7)**

In MAP(7), the distance matrix from specialized GMMs were used for experiments. Normally, its performance was less than 38% indicating that many specialized GMMs were not well built. However, DTSVM helped to increase the performance from the lowest 32.91% to 46.61%. This result strongly shows the importance of exploring the optimal coefficients of base kernels.

	SVM_T	**SVM_AT**	**FR**	**A-SVM**	**MKL**	**DTSVM**	**A-MKL**
one-vs-all	49.94 ± 6.96	57.85 ± 1.91	57.18 ± 8.59	48.25 ± 8.43	58.53 ± 2.36	55.82 ± 0.83	$\mathbf{58.87 \pm 0.90}$
one-vs-one	33.67 ± 14.67	51.64 ± 0.45	57.18 ± 6.21	36.05 ± 12.62	49.94 ± 1.05	50.85 ± 0.80	55.14 ± 2.10

Table 16: Means and standard deviations (percent) of average recognition accuracies over six events for all methods using different SVM multi-class schemes

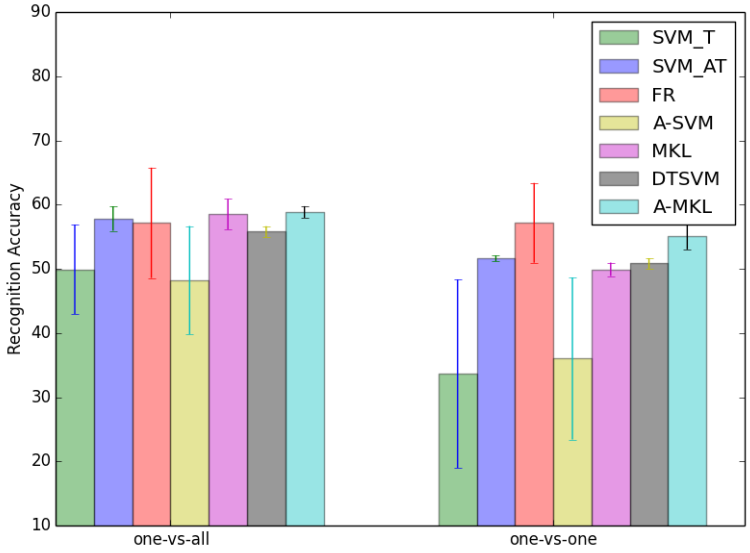

Figure 17: Chart representation of Table 16

	SVM_T	**SVM_AT**	**FR**	**A-SVM**	**MKL**	**DTSVM**	**A-MKL**
one-vs-all	0.98	8.54	10.53	12.52	8.71	11.33	27.87
one-vs-one	1.49	4.08	5.68	7.16	3.44	5.05	10.94

Table 17: Average running time (seconds) for all methods using different SVM multi-class schemes

In previous experiments, MAP acted as the performance metric. However, in real applications, the classifier is expected to output labels rather than scores. Based on these predictions, the recognition accuracies of classifiers could be easily calculated. In the experiments shown in Table 16 and 17, recognition accuracy acts

as the performance metric. There are generally two approaches to enable SVM to perform multi-class classification: one-vs-one and one-vs-all. For one-vs-one, if there are n classes, $n*(n-1)/2$ classifiers are built by taking training data from any two classes. For instance, there are in total 6 classes. Thus, 15 classifiers are built because of 15 unique combinations. Finally, the prediction for each sample is chosen to be the class with the highest vote. On the contrary, one-vs-all only trains n classifiers if there are n classes. For each classifier on each class, the training data is formed by taking the current class as positive class while the rest of classes are treated as negative class. For instance, in this experiment, the training data for "birthday or non-birthday" classifier is constructed by taking all "birthday" samples as positive samples while the rest samples as negative samples. Finally, the prediction of each testing sample is chosen to be the class with the highest SVM score. In this experiment, these two different multi-class SVMs were implemented. Following the same settings as before, the experiment was repeated for five times, and the accuracies and average running time are recorded in Table 16 and Table 17 respectively.

According to these two tables, there are two observations.

1. **In this case, one-vs-all constantly performed better than one-vs-one**

   The possible reason may be a lack of training samples for one-vs-one. When building classifiers in one-vs-one manner, the training samples are divided based on different class combinations. However, for one-vs-all, the training samples are the whole available training samples. Moreover, as expected, A-MKL performed the best.

2. **In this case, one-vs-all required more running time compared with one-vs-one**

   Again, since one-vs-all deals with more training samples, it costs more time to build classifiers. Also, it is worth noting that A-MKL took 27.87 seconds to finish the prediction task. However, compared with SVM_AT, the performance was only increased by 1.02%. There is a trade-off between accuracy and running time. Thus, it depends on the system requirements to choose which implementation.

7 Web-based Demo System

In order to better illustrate the work that has been done in this final year project, a web-based demo system is also implemented based on Django, which is a popular python web framework. There are generally two components for web application: back end and front end, where front end is responsible to collect data from users and process it to conform the requirements of back end, and back end processes the received data. Back to this demo system, Django adopts previous python modules like extracting SIFT features and calculating EMD distance as the back end side. For the front end, a CSS library called Bootstrap is used for better visualization. Also, ajax is heavily used to connect the user input with the back end server and dynamically update the web page.

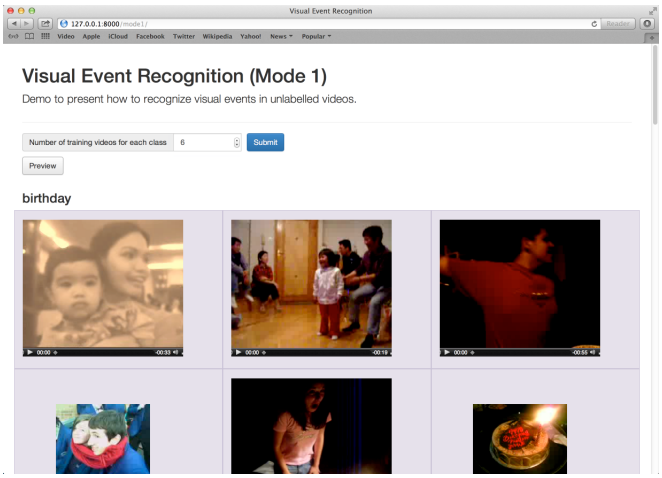

Figure 18: Snapshot of mode 1. In mode 1, the distance matrix of Youtube videos is calculated offline. The user could randomly select the training data and testing data to check the recognition performance.

There are two modes built for two different purposes as shown in Figure 18 and 19, respectively.

- **Mode 1**

  Mode 1 enables the user to randomly select training and testing Youtube videos for recognition. The distance matrix used in recognition is calculated offline to save time. The procedures go as following. Firstly, the user selects the number of videos per class as training data. Secondly, the classifier based on the training data is built. Thirdly, recognitions on the rest of videos are performed using the classifier built in the previous step. Lastly, in the result section, not only the recognition accuracy is displayed, but also a confusion matrix, which better visualizes the performance.

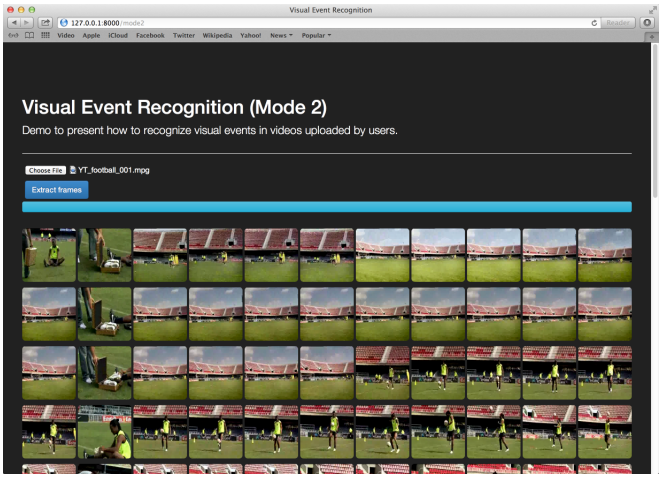

Figure 19: Snapshot of mode 2. In mode 2, the user could upload a new video and get the label of this video predicted by the underlying recognition system

- **Mode 2**

  The second mode aims to present users the whole process of recognizing videos. The user could upload a new video and check how this video is recognized from extracting frames, extracting features and building histograms all the way down to calculating distances with training videos and building a classifier for recognition. Such interactive presentation shall give users a clear idea about video recognition.

To conclude, this web-based demo system presents visual event recognition in two modes nicely. Following the procedures, users shall be able to understand the underlying operations to recognize videos.

8 Conclusion

In conclusion, a recognition system, which recognizes images and videos, has been successfully implemented in various approaches. In image recognition, spatial pyramid is adopted to represents images, and such representations of videos are later on input into SVM and KNN for classification. Moreover, earth mover's distance is incorporated into the calculation of image-to-image distances, and the experiments demonstrate the robustness of EMD. For video recognition, which is the main objective of this project, bag of words and specialized GMMs are used to represent videos compactly, and respective distance calculation methods are employed to measure video-to-video distances. Once the distance calculation is done, it is then converted into gram matrix using four different types of kernel, and SVM is employed to perform classification. Furthermore, four different domain adaptation methods are studied and implemented to improve the performance of classifiers by leveraging labelled samples from other domains. The best result is achieved by Adaptive Multiple Kernel Learning (A-MKL) through selecting two distance matrices smartly, and the recorded mean average precision 61.4% is better than that reported in the studied paper [11]. Comprehensive experiments have been conducted on these various approaches, and detailed analysis is also included. Last but not least, a web-based demo system is built to better present the work in this project.

As for future recommendations, there are three suggestions. The first one is to incorporate more types of features including space-time feature and acoustic feature. With more features extracted from videos, the recognition system is expected to be more robust. Secondly, more concept attribute detectors could be employed to represent videos with more attributes. In this way, the produced attribute vector shall possess more discriminative power. Lastly, various domain adaptations could be combined wisely. For instance, Feature Replication could be incorporated into Adaptive Multiple Kernel Learning.

References

[1] A. Agarwal and B. Triggs, "Hyperfeatures–multilevel local coding for visual recognition," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 30–43.

[2] M. Baillie and J. M. Jose, "Audio-based event detection for sports video," in *Image and Video Retrieval*. Springer, 2003, pp. 300–309.

[3] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel for image classification," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3. IEEE, 2003, pp. III–513.

[4] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[6] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[7] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[9] H. Daumé III, "Frustratingly easy domain adaptation," in *ACL*, vol. 1785, no. 1786, 2007, p. 1787.

[10] L. Duan, I. W. Tsang, D. Xu, and S. J. Maybank, "Domain transfer svm for video concept detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1375–1381.

[11] L. Duan, D. Xu, I. W.-H. Tsang, and J. Luo, "Visual event recognition in videos by learning from web data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1667–1680, 2012.

[12] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1458–1465.

[13] C.-L. Huang, H.-C. Shih, and C.-Y. Chao, "Semantic analysis of soccer video using dynamic bayesian network," *Multimedia, IEEE Transactions on*, vol. 8, no. 4, pp. 749–760, 2006.

[14] Y.-G. Jiang, S. Bhattacharya, S.-F. Chang, and M. Shah, "High-level event recognition in unconstrained videos," *International Journal of Multimedia Information Retrieval*, pp. 1–29, 2012.

[15] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *Proceedings of the 6th ACM international conference on Image and video retrieval*. ACM, 2007, pp. 494–501.

[16] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *The Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[17] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[18] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[19] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2169–2178.

[20] P. Li, J. Ma, and S. Gao, "Actions in still web images: Visualization, detection and retrieval," in *Web-Age Information Management*. Springer, 2011, pp. 302–313.

[21] T. Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[22] J. Liu, Q. Yu, O. Javed, S. Ali, A. Tamrakar, A. Divakaran, H. Cheng, and H. S. Sawhney, "Video event recognition using concept attributes." in *WACV*, 2013, pp. 339–346.

[23] S. Lloyd, "Least squares quantization in pcm," *Information Theory, IEEE Transactions on*, vol. 28, no. 2, pp. 129–137, 1982.

[24] A. Loui, J. Luo, S.-F. Chang, D. Ellis, W. Jiang, L. Kennedy, K. Lee, and A. Yanagawa, "Kodak's consumer video benchmark data set: concept definition and annotation," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 2007, pp. 245–254.

[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[26] B. S. Manjunath and W.-Y. Ma, "Texture features for browsing and retrieval of image data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 8, pp. 837–842, 1996.

[27] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International journal of computer vision*, vol. 60, no. 1, pp. 63–86, 2004.

[28] P. Natarajan, P. Natarajan, V. Manohar, S. Wu, S. Tsakalidis, S. N. Vitaladevuni, X. Zhuang, R. Prasad, G. Ye, D. Liu *et al.*, "Bbn viser trecvid 2011 multimedia event detection system," in *NIST TRECVID Workshop*, 2011.

[29] A. Natsev, J. R. Smith, M. Hill, G. Hua, B. Huang, M. Merler, L. Xie, H. Ouyang, and M. Zhou, "Ibm research trecvid-2010 video copy detection and multimedia event detection system," in *Proceedings of NIST TRECVID, Workshop*, 2010.

[30] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, pp. 971–987, 2002.

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

[32] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1, pp. 19–41, 2000.

[33] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[34] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[35] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1177–1178.

[36] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[37] J. E. Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. O'Reilly, 2012.

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

[39] M. Wang, R. Hong, G. Li, Z.-J. Zha, S. Yan, and T.-S. Chua, "Event driven web video summarization by tag localization and key-shot identification," *Multimedia, IEEE Transactions on*, vol. 14, no. 4, pp. 975–985, 2012.

[40] L. Xie, S.-F. Chang, A. Divakaran, and H. Sun, "Structure analysis of soccer video with hidden markov models," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 4. IEEE, 2002, pp. IV–4096.

[41] D. Xu and S.-F. Chang, "Visual event recognition in news video using kernel methods with multi-level temporal alignment," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[42] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification," in *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 2007, pp. 197–206.

[43] J. Yang, R. Yan, and A. G. Hauptmann, "Cross-domain video concept detection using adaptive svms," in *Proceedings of the 15th international conference on Multimedia*. ACM, 2007, pp. 188–197.

[44] W.-L. Zhao, C.-W. Ngo, H.-K. Tan, and X. Wu, "Near-duplicate keyframe identification with interest point matching and pattern learning," *Multimedia, IEEE Transactions on*, vol. 9, no. 5, pp. 1037–1048, 2007.

[45] X. Zhou, X. Zhuang, S. Yan, S.-F. Chang, M. Hasegawa-Johnson, and T. S. Huang, "Sift-bag kernel for video event analysis," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2008, pp. 229–238.