1 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 [11], 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.

1.1 Framework description

Generally, there are 4 below steps to go:

1.1.1 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 [9]. On the other hand, local features refer to features extracted from local patches (or interest points). Popular local feature extraction methods include Harris-Laplace [5], Hessian [10], Scale Invariant Feature Transform [7] and so forth. Among popular local features, *Scale Invariant Feature Transform (SIFT)*, introduced by David Lowe in [7], is one of the most successful local image descriptors in the last decade. Because of the robustness of SIFT shown in various literatures [7, 2, 17], SIFT is chosen to be the feature to represent 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*:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

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 candidates locations are filtered to remove unstable points based on criteria including low contrast and edge response. Please refer to the paper [8] 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 4 below demonstrates the whole process to construct SIFT descriptor.

For simplicity, an open source library VLFEAT [14] is adopted to extract SIFT from images. A typical image of size 200×200 normally produces around 800 SIFT features.

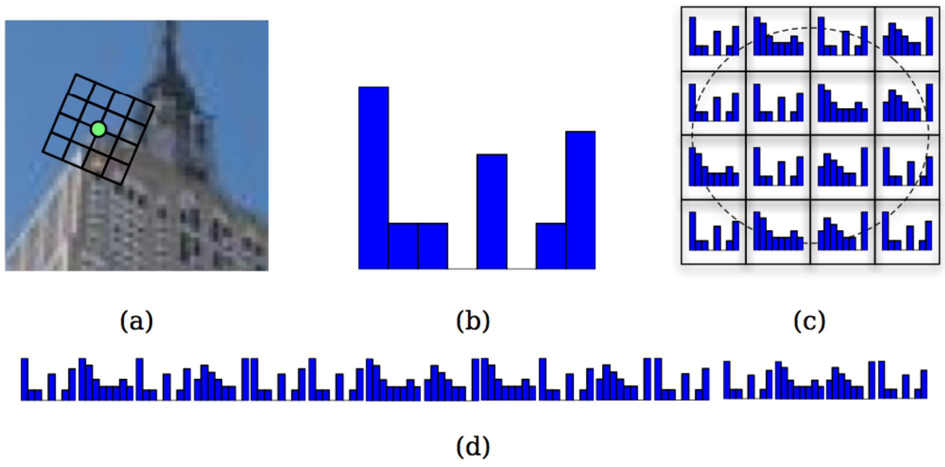

Figure 1: SIFT descriptor [13] (a) a frame with 4 × 4 subregions around an interest point. (b) an 8 bin histogram over the direction of gradient in one subregion. (c) all histograms in their respective subregions. (d) all 16 histograms are concatenated to form a 128 (16 × 8) dimensional vector.

1.1.2 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 causes 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 [6] 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, better method is needed to build up vocabularies. Mini-Batch K-Means [12], a variant of KMeans algorithm which uses mini-batches to reduce the computation time, is a good alternative. In contrast to other algorithms which reduces 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

```
1: Given: $k$, mini-batch size $b$, iteration $t$, data set $X$
2: Initialize each $c \in C$ with an $x$ picked randomly from $X$
3: $v \leftarrow 0$
4: for $i = 1$ to $t$ do
5:     $M \leftarrow b$ examples picked randomly from $X$
6:     for $x \in M$ do
7:         $d[x] \leftarrow f(C, x)$                ▷ Cache the nearest center to $x$
8:     end for
9:     for $x \in M$ do
10:        $c \leftarrow d[x]$                     ▷ Get cached center for this $x$
11:        $v[c] \leftarrow v[c] + 1$              ▷ Update per-center counts
12:        $\eta \leftarrow \frac{1}{v[c]}$        ▷ Get per-center learning rate
13:        $c \leftarrow (1 - \eta)c + \eta x$     ▷ Take gradient step
14:    end for
15: 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.

1.1.3 Construct a pyramid of three levels for each image

1.1.4 Classify based on above representations

1. **Construct a pyramid of three levels for each image**
   The pyramid match kernel is introduced by Kristen Grauman and Trevor Darrell [3] in 2005. In this method, a pyramid is built for each image level

by level through increasing histogram resolution. However, this method ignores spatial information among images. After one year, spatial pyramid matching is introduced [4]. Like the name has stated, pyramid is built by taking spatial information into consideration. There are three levels built: level 0 takes the whole image as input and produce a single histogram; level 1 divides an image into 4 parts equally and produces 4 histogram in total; level 2 divides an image into 16 parts equally and outputs 16 histograms in total. Also, the weight is decreasing with level increasing when calculating distances between images. In this experiment, the spatial pyramid matching is implemented independently without help from libraries. Since the vocabulary size is 300, the numbers of dimension are 300, 1500, 6300 for $Level1$, $(Level1 + Level2)$ and $(Level1 + Level2 + Level3)$ respectively.

2. **Classify based on above representations**
   Support vector machines are supervised learning models to analyze data and recognize patterns. In addition to performing linear classification, SVMs can also efficiently perform non-linear classification through kernel trick, which implicitly maps data into high dimensional feature spaces. In this experiment, LIBSVM [1] is used, and multi-class classification is performed in "one-against-one" approach. Experiments of different kernels at different levels have been conducted, and detailed analysis are presented in experiment section.

1.2 Earth mover's distance and its application

According to Rubner, Tomasi and Guibas [11], the earth mover's distance is formulated as the following linear programming problem: Let $P = \{(p_1, w_{p1}), ..., (p_m, w_{pm})\}$ be the first signature with m clusters, where p_i is the cluster representative and w_{pi} is the weight of the cluster; $Q = \{(q1, w_{q1}), ..., (qn, w_{qn})\}$ 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} the flow between p_i and q_j, that minimizes the overall cost:

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij},$$

subjective to the following constraints:

$$f_{ij} \geq 0 \qquad 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^{n} f_{ij} \leq w_{pi} \quad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{qj} \quad 1 \leq j \leq n$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj})$$

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

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

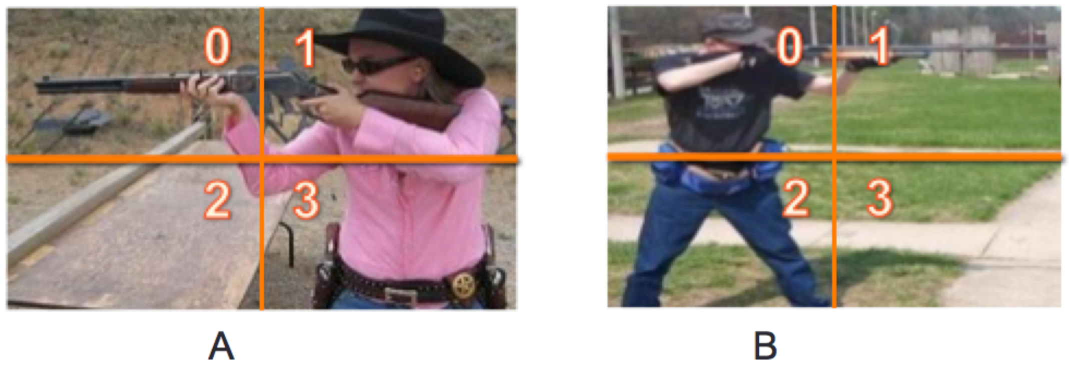

Figure 2: Image division

Specific to calculate image distances, the first step is to divide each image into four parts equally. An example is depicted in Figure 1. There are two images A and B, and they are both divided into 4 parts. Each part is then represented by a histogram using the vocabulary built above. As a result, such representation is identical to pyramid at level one. Afterward, image A and B are represented as below:

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

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

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

Here, the distance matrix D is constructed by Euclidean distance between histograms. A sample distance matrix of the two images in Figure 1 is shown below in Table 1.

	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 1: Distance matrix

If there is no alignment, the distance is calculated as below:

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

Once the linear programming is solved, the matches depicted in Figure 2 are found. In this case, the EMD is calculated as below:

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

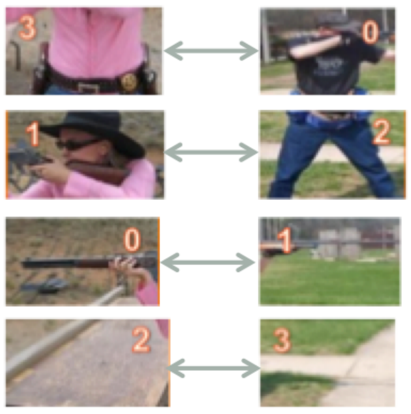

Figure 3: EMD match of images

It is easy to see that EMD is significantly smaller than the distance calculated without alignment. As Figure 2 shows, a_0 and b_1, which both represent a gun, are perfectly matched. Such example shows the effectiveness of earth mover's distance. More details about EMD's application in image recognition are presented in the experimental section.

1.3 Key frame identification in cluster of images

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 three images are examined to be near duplicates, only one image will be selected to represent these three images. The rest of this section will focus on two parts: near duplicate identification and how this duplicate identification is applied to identify key frames in videos.

1.3.1 Identifying near duplicate

Currently, the way which has been implemented to check whether two images are near duplicate follows the method stated on paper [16]. In the paper, there are basically three steps. 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 now, and a threshold is set to replace the classifier. It means that two images are treated as near duplicate as long as the number of matching interest point is large enough. Later on, if possible, a classifier will be built to enhance the performance.

In order to minimize false matches, the authors propose one-to-one symmetric matching [16], 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 3. Given each point $P = [p_1, p_2, ..., p_{36}]$ of I_1, the index of p_i is hashed to,

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

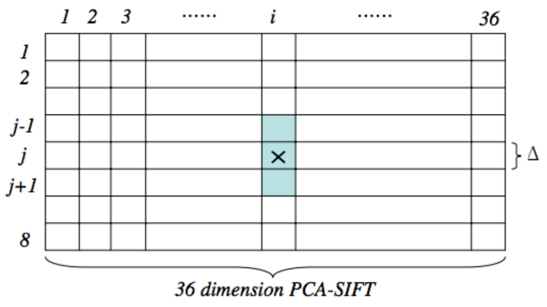

Figure 4: 8×36 Hash Table [16]

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 4. 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. Please also notice that only partial matching points are drawn for better visualization.

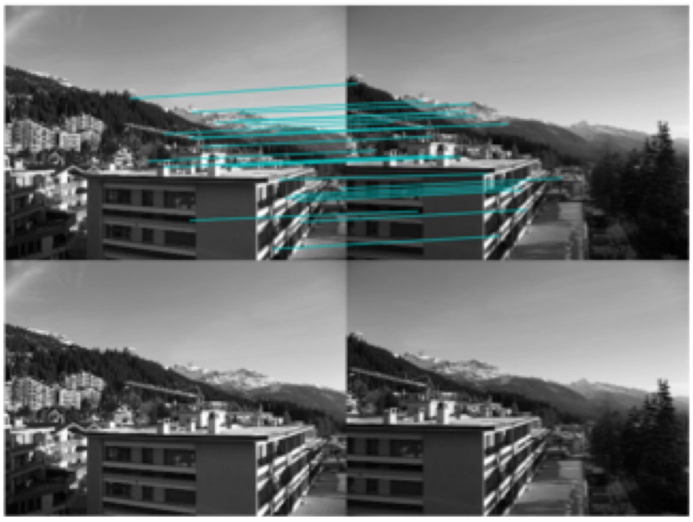

Figure 5: Near duplicate example

1.3.2 Identifying key frames

Now that near duplicate method is introduced, it is time to check how this method is 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 [15], it is recommended to perform K-means algorithm on all the frames at first. Later on, NDK 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, 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 of selected to be a key frame. If there is a tie, the key frame is then randomly chose among candidates.

A very good example is illustrated in Figure 5. These frames are sampled at a rate of one frame per second from a video introducing how to use Google glass. Figure 5 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 video. Because all similar frames are discarded, the three resulting frames are indeed representative.

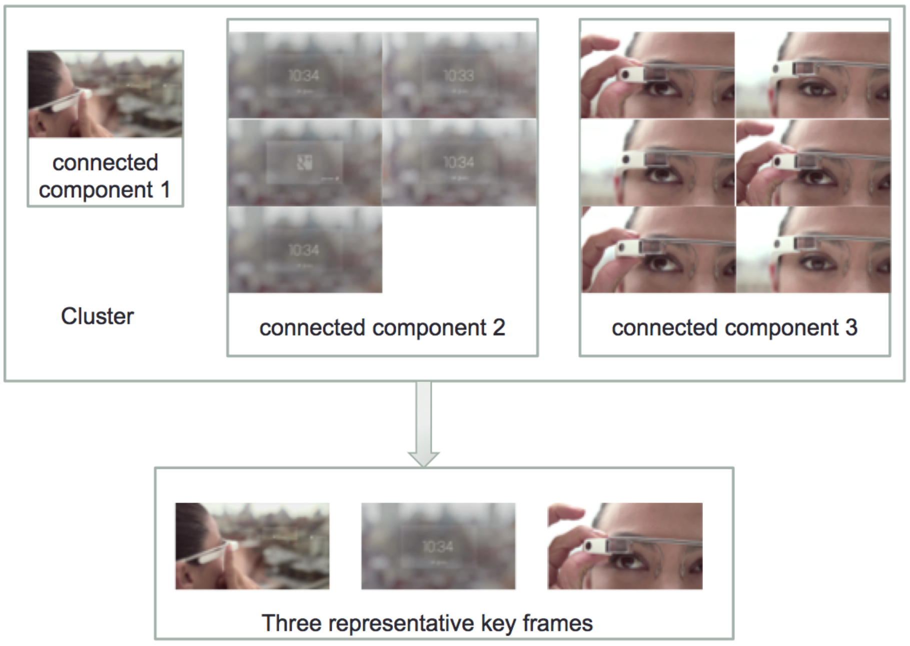

Figure 6: Key frames identification

2 Video recognition

As introduced in the section Image Recognition, recognition tasks could be simplified into a problem to calculate distances between different items 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 of each video clip is needed.

2.1 Representations of videos

2.1.1 Bag of words

2.1.2 Gaussian mixture models

2.2 Distance calculation

2.2.1 Aligned space-time pyramid matching

2.2.2 Distances between Gaussian mixture models

2.3 Kernels for classification

2.4 Other approach: concept attributes

3 Domain adaption

3.1 Overview

3.2 Feature replication (FR)

3.3 Adaptive support vector machine (A-SVM)

3.4 Multiple kernel learning (MKL)

3.5 Domain transfer support vector machine (DTSVM)

3.6 Adaptive multiple kernel learning (A-MKL)

4 Experiments

5 Web-based demo system

5.1 Design

5.2 Implementation

6 Conclusion

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. IEEE, 1999, pp. 1150–1157.

[8] ——, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.