

Hybrid negative example selection using visual and conceptual features

Kimiaki Shirahama · Yuta Matsuoka · Kuniaki Uehara

Published online: 15 October 2011

© The Author(s) 2011. This article is published with open access at Springerlink.com

Abstract An application of Query-By-Example (QBE) is presented where shots that are visually similar to provided example shots are retrieved. To implement QBE, counter-example shots are required to accurately distinguish shots that are relevant to the query from those that are not (Li and Snoek (2009), Yu et al. (2004)). However, there are usually a huge number of shots, not relevant to a particular query, which can serve as counter-example shots. It is difficult for a user to provide counter-example shots that would aid retrieval. Thus, we developed a QBE method based on *partially supervised learning* where a retrieval model is constructed by selecting counter-example shots from shots without user supervision. To ensure the speed and accuracy of the QBE method, we select a small number of counter-example shots that are visually similar to given example shots but irrelevant to the query. Such shots are useful for characterizing the boundary between relevant and irrelevant shots. For our method, we first filter shots that are visually dissimilar to example shots based on SVMs on a visual feature. Then we filter shots relevant to the query based on concept detection results from pre-constructed classifiers. Shots that pass the above two tests are considered as counter-example shots. Experimental results obtained using TRECVID 2009 video data validate the effectiveness of our method.

Keywords Negative example selection · Partially supervised learning · Query by example · Visual feature · Conceptual feature

K. Shirahama (✉)

Graduate School of Economics, Kobe University, 2-1, Rokkodai, Nada, Kobe 657–8501, Japan
e-mail: shirahama@econ.kobe-u.ac.jp

Y. Matsuoka

Graduate School of Engineering, Kobe University, 1-1, Rokkodai, Nada, Kobe 657–8501, Japan
e-mail: matuoka@ai.cs.scitec.kobe-u.ac.jp

K. Uehara

Graduate School of System Informatics, Kobe University, 1-1, Rokkodai, Nada, Kobe 657–8501, Japan
e-mail: uehara@kobe-u.ac.jp

1 Introduction

With the rapidly increasing amount of video data available on the internet, it has become important to develop a video retrieval method that can efficiently retrieve shots relevant to a query. Based on how the query is represented, existing video retrieval methods can be classified into two types, *Query-By-Keyword* (QBK) and *Query-By-Example* (QBE). In QBK, the user represents the query using keywords, and shots annotated with the same or similar keywords are retrieved. In QBE, the user provides example shots that represent the query, and then shots are retrieved based on their similarity to example shots in terms of visual features. QBE has the following advantages over QBK. QBE is effective as the query is objectively represented by visual features in example shots. In contrast, when a query consists of keywords, it is often difficult to select appropriate keywords for relevant shots due to lexical ambiguity and user subjectivity. Furthermore, since QBE uses features extracted automatically from shots, no shot annotation is required. In other words, as long as example shots are provided, QBE should work for any query. This paper aims to improve on QBE, while keeping the above advantages in mind.

QBE can be formulated as a classification problem in machine learning, where example shots are used to construct a classifier that classifies shots as relevant or irrelevant to a query. In this formulation, example shots are *positive examples* representing relevant shots. However, *negative examples*, i.e., shots irrelevant to the query, are not provided. Thus, one-class classification methods, such as Nearest Neighbor and one-class Support Vector Machine (SVM) [22], appear suitable for QBE. However, in one-class classification, a classifier is constructed to distinguish positive examples from all the other examples. This means that a boundary between relevant and irrelevant shots is supported only from the positive side. In other words, one-class classification simply extracts a dense region of positive examples from the visual feature space. Therefore, while a large number of positive examples are required to extract a generalized region, it is impractical for a user to provide so many positive examples. Many research papers have reported that the performance of one-class classification methods is considerably inferior to two-class classification methods that use both positive and negative examples [10, 28]. Therefore, negative examples are essential for accurate retrieval using QBE.

We select negative examples from the set of shots that are not positive examples. Such shots that have no class labels are called *unlabeled examples*. This approach of constructing a classifier using positive and unlabeled examples is known as *Partially Supervised Learning* (PSL) [5, 6, 11, 28]. Thus, we formulate QBE as a PSL problem. One of the most important issues in a PSL problem is to construct a classifier. In video retrieval, each example is generally represented by a high-dimensional feature. For example, one of the most popular representations is a ‘Bag-of-Visual-Words’ (BoVW) with over a thousand dimensions, in which each dimension represents the frequency of a local edge shape [7, 13, 15, 20]. For such a high-dimensional feature, it has been established that an SVM [25] is one of the most effective classifiers, as it extracts a robust decision boundary between positive and negative examples based on the margin maximization principle. Furthermore, a complex (non-linear) decision boundary can be extracted using a non-linear SVM. In this process, examples in the high-dimensional feature space are mapped into a higher-dimensional feature space using a kernel trick. For these reasons, we use the non-linear SVM as a classifier in the PSL problem.

However, the computational cost of a non-linear SVM is $O(n^3)$ where n is the number of positive and negative examples [24]. Given positive examples, existing PSL methods such

as [5, 6, 11, 28] select a large number of negative examples, without considering the usefulness of each negative example. It should be noted that SVM classification only requires *support vectors*, which are the positive or negative examples closest to the decision boundary, while all other examples are redundant. Therefore, for fast and accurate SVM classification, we develop a PSL method that can select a small number of negative examples which are likely to become support vectors. The selected negative examples should be visually similar to positive examples but irrelevant to the query.

We select such negative examples based on *visual* and *conceptual* features. A visual feature such as color, edge or motion represents a particular visual characteristic of an example. Such a feature can be automatically extracted with no prior knowledge. A conceptual feature represents the presence of a concept, like *Person*, *Car* or *Building*, and the detection of such a feature requires prior knowledge. Much of the recent research has focused on automatic concept detection, where a detector for each concept is constructed using large amounts of training data [13, 20, 23]. Concept detection results are used not just to detect the presence of a concept, but also as features for video retrieval. Each example is represented as a vector, where each dimension represents the detection score for a concept. This score represents the likelihood of whether the concept is present in the example. In TRECVID, an annual worldwide competition for video retrieval techniques [19], the effectiveness of concept detection results has been proven, as almost all top-ranked video retrieval methods use concept detection scores [20, 31]. In the following discussion, we use the term ‘conceptual feature’ for a vector consisting of concept detection scores.

Although a conceptual feature may initially appear to be much more meaningful than a visual feature, both features are required to select useful negative examples. Recall that the objective of QBE is to construct a classifier (retrieval model) on the visual feature. Thus, negative examples visually similar to positive examples are required to extract an appropriate boundary between relevant and irrelevant shots. Let us consider the query “a building is shown”, and a shot showing a closet. This shot is clearly irrelevant to the query, but is characterized by a rectangular shape similar to a building. Therefore, the shot is likely to be a negative example useful for characterizing the boundary between relevant and irrelevant shots. However, since concept detectors are precisely tuned using large amounts of training data, the conceptual feature of the shot may be significantly different to positive examples. In other words, even when unlabeled examples are visually similar to positive examples, the difference in the concepts that appear in them means they have significantly different conceptual features. Thus, we use visual features to evaluate whether unlabeled examples are visually similar to positive examples.

Unlabeled examples can be either relevant or irrelevant to the query. If such examples are used as negative examples, the resulting classifier will incorrectly classify many relevant shots as irrelevant. Therefore, unlabeled examples relevant to the query should be filtered. However, in this case, visual features should not be used due to the insufficient number of positive examples provided by a user. Even if certain unlabeled examples are relevant to the query, they may have been shot using different camera techniques and settings compared to the positive examples. As a result, several other unlabeled examples, irrelevant to the query, may be more similar to positive examples. Hence, we use conceptual features to filter unlabeled examples relevant to the query. As concept detectors are constructed using large amounts of training data, concepts related to the query can be robustly detected independently of their sizes, directions and positions on the screen.

Based on this discussion, we developed the PSL method illustrated in Fig. 1, where circles and question marks represent positive and unlabeled examples, respectively. Given

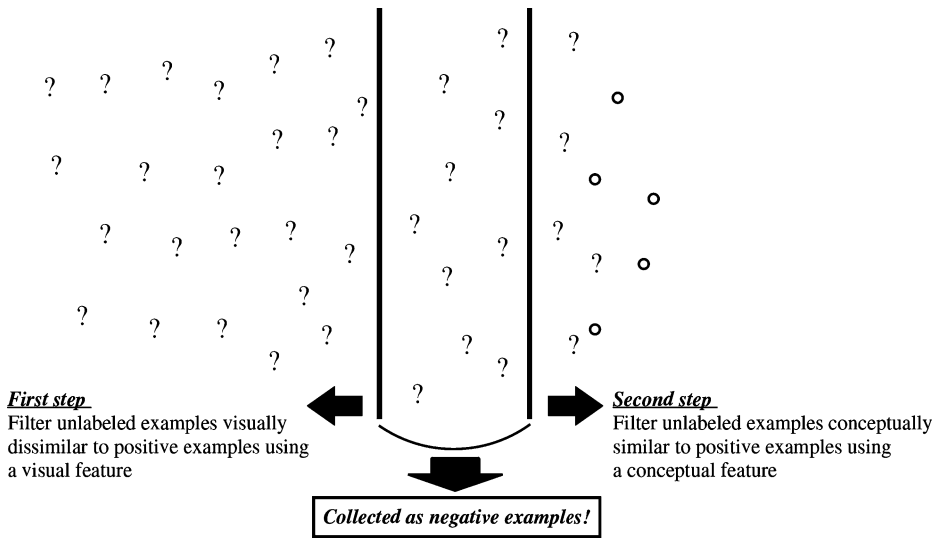


Fig. 1 An illustration of our PSL method

positive examples for a query, our PSL method takes two steps to find negative examples that are visually similar but irrelevant to the query. We first filter unlabeled examples that are visually dissimilar to positive examples using a visual feature (Fig. 1, left). Next, we filter unlabeled examples relevant to the query, selecting examples that are similar to positive examples in terms of a conceptual feature (Fig. 1, right). Finally, unlabeled examples that pass the above two tests are labeled as negative examples, and are used to construct a retrieval model (classifier) in QBE.

2 Related works

In the field of image/video annotation and retrieval, many researchers have simply used randomly selected unlabeled examples as negative examples [12, 13, 18, 20]. However, there is no guarantee that the use of such negative examples will lead to accurate retrieval. Yan et al. proposed a method which uses unlabeled examples that are the most dissimilar to positive examples as negative examples [26]. However, these are of no use for determining the boundary between relevant and irrelevant shots. Tešić et al. proposed a negative example selection method that uses a conceptual feature [23]. In order to select a diverse set of negative examples, they group unlabeled examples into clusters based on the conceptual feature, and select negative examples as cluster centers. However, selecting only cluster centers as negative examples may exclude many useful negative examples. Li et al. proposed a method that selects negative examples from a tagged image collection on a social site, such as Flickr and Facebook [10]. They first filter images tagged with synonyms of query words, and negative examples are then selected as randomly sampled images. In comparison, our PSL method selects negative examples without any human intervention. To the best of our knowledge, no existing method uses both visual and conceptual features,

and selects negative examples visually similar to positive examples but irrelevant to a query.

In the field of machine learning, many researchers have studied PSL [5, 6, 11, 28]. For example, Liu et al. [11] proposed a method that selects some positive examples as ‘spy’ examples and adds them to the set of unlabeled examples. A naive Bayesian classifier is then built using positive and unlabeled examples, where these spy examples are used to set the probabilistic threshold for considering unlabeled examples as negative examples. Yu et al. [28] proposed a method that iteratively collects negative examples. Over each iteration, first an SVM is built based on positive examples and already selected negative examples. Then, unlabeled examples classified as negative by the SVM are selected as new negative examples.

Fung et al. [6] proposed a method that selects both positive and negative examples among unlabeled examples. Negative examples are selected iteratively as unlabeled examples that are significantly similar to already selected negative examples. Afterwards, positive examples are selected as unlabeled examples that are significantly similar to already selected positive examples. Elkan et al. proposed a method which builds a classifier by assigning probabilistic weights to unlabeled examples [5]. It should be noted that all the above methods simply assign labels or weights to a large number of unlabeled examples. Thus, a large amount of computational time is needed to construct an SVM.

The selection of more useful examples has been investigated in *active learning*, which is an interactive learning method whereby a system selects unlabeled examples useful for improving classification performance, and asks a user to label them [16]. Good classification performance is achieved while manual labeling work is minimal. In active learning, one popular heuristic is *uncertainty sampling*, which selects unlabeled examples close to the decision boundary of the current classifier. By labeling such examples, the decision boundary can be updated appropriately. Unlike active learning, our PSL method requires no human intervention.

The selection of more useful examples has also been studied in the context of the *class imbalance* problem [1]. This problem occurs when building a well-generalized classifier becomes difficult because the number of examples in the majority class is much more than in the minority class. In QBE, positive and negative examples constitute the minority and majority classes, respectively, as the number of relevant shots is generally much less than the number of irrelevant shots. In such a case, the simplest hypothesis of classifying all examples as negative proves to be reasonably accurate on training examples (i.e. positive and negative examples), but it is clearly of no use for classifying unseen examples. Thus, to balance positive and negative examples, methods for over-sampling positive examples and under-sampling negative examples have been proposed. For over-sampling positive examples, Akbani et al. [1] and Peng et al. [14] used the Synthetic Minority Oversampling TEchnique (SMOTE) which synthetically generates new positive examples between existing positive examples. For under-sampling negative examples, Peng et al. [14] proposed a method which selects a small number of diverse negative examples, where each negative example is selected from a different cluster. In addition, Yuan et al. [29] proposed a method which iteratively filters negative examples dissimilar to positive examples, by building SVMs using positive examples and the remaining negative examples. The negative examples that remain are useful for characterizing the boundary between relevant and irrelevant shots. It should be noted that the above methods only work on

labeled examples, while in QBE only a small number of positive examples are labeled, while the rest are unlabeled.

Our PSL method is an extension of the method in [29] to QBE. Specifically, based on the method in [29], we iteratively filter unlabeled examples visually dissimilar to positive examples by building SVMs using positive examples and the remaining unlabeled examples. However, the unlabeled examples that remain include both relevant and irrelevant shots. Thus, we filter unlabeled examples relevant to the query using a conceptual feature. Note that these unlabeled examples should not be used as positive examples because accurate detection of positive examples is difficult even with the help of the conceptual feature.

In addition, our preliminary experiment showed that the class imbalance problem is not very important in QBE. This experiment was conducted on TRECVID 2009 video data, consisting of 36,106 and 97,150 shots in 219 development and 619 test videos, respectively [19]. For each query in the top row in Table 1, we manually select positive examples from development videos. The number of positive examples is given in parentheses under the query name. In the second and third rows in Table 1, we compare two types of retrievals, *All* and *Random*. In *All*, an SVM is constructed by regarding all the unlabeled examples as negative examples. The number of such negative examples is shown in parentheses in the second row. We can see the imbalance ratio between the positive and negative examples where there are over 200 times more negative examples than positive examples. In *Random*, we balance positive and negative examples. The numbers within parentheses in the third row indicate that the number of negative examples that are randomly selected among unlabeled examples, is ten times greater than the number of positive examples [23]. Based on the above, *All* and *Random* retrieve relevant shots in test videos by constructing SVMs using the same set of positive examples and different sets of negative examples. The second and third rows represent the number of relevant shots among 1,000 shots, for which the SVMs in *All* and *Random* assign the highest probabilities of being relevant. Note that the third row represents the average of 10 retrieval results by *Random* using different sets of randomly selected negative examples.

As can be seen from Table 1, *All* outperforms *Random* for all queries. This means that even if there is an imbalance between positive and negative examples, using more negative examples leads to more accurate retrieval. One reason for this is that the class imbalance problem is only significant with linearly non-separable data [29]. Each example is represented using a high-dimensional feature. In this experiment, each example is represented using a 1,000 dimensional BoVW representation. The number of examples required to fill such a high-dimensional feature space is exponentially larger than the number of dimensions. Therefore, even if tens of thousands of examples are used, their distribution will be very sparse and they will be linearly separable. One objective of our PSL method is to collect a small number of negative examples for comparable or even more accurate retrieval than *All*.

Table 1 Performance comparison between the class imbalance case (*All*) and the balance case (*Random*)

Query	Crowd (97)	Tall building (100)	Computer (61)	Airplane/helicopter (40)	Talking (124)
All	212 (36,009)	175 (36,006)	140 (36,045)	46 (36,066)	170 (35,982)
Random	204.5 (970)	128.6 (1,000)	133.2 (610)	36.9 (400)	149.3 (1,240)

3 Negative example selection using visual and conceptual features

In this paper, we target queries where the identification of object categories is important for retrieving relevant shots, such as “tall buildings are shown” and “people appear with computers”. We use a *SIFT* (Scale-Invariant Feature Transformation) as a visual feature. The effectiveness of *SIFT* features in identifying object categories has been proven in many previous works [7, 15, 30]. A *SIFT* feature represents the local shape around an interest point detected in an image. We represent each shot using a BoVW representation on the *SIFT* feature [7, 15, 30]. In the BoVW, many *SIFT* features are first grouped into clusters, where each cluster center represents a characteristic local shape, called a ‘visual word’. Then, each *SIFT* feature extracted from a shot is assigned to the most similar visual word. As a result, the shot is represented by a vector where each dimension represents the frequency of a visual word. In this paper, using the software developed by Sande et al. [15], we represent each shot with the following BoVW.

1. Interest points in the *keyframe* of each shot are detected by Harris-Laplace detector. We define the *keyframe* as the middle video frame in the shot. This is because the semantic content is spatially and temporally continuous in the shot. Therefore, we assume that the representative semantic content is shown in the *keyframe*. Each detected interest point is described by a *SIFT* feature.
2. Group randomly sampled 200,000 *SIFT* features into 1,000 clusters using the *k*-means clustering algorithm. Each cluster center is regarded as a visual word.
3. Assign each *SIFT* feature extracted from a shot to the most similar visual word. As a result, the shot is represented as a 1,000-dimensional vector.

In process 3, to avoid assigning similar *SIFT* features to different visual words, we use the ‘soft assignment’ approach, which smooths the distribution of visual words based on kernel density estimation with a Gaussian kernel [15].

For a conceptual feature, we use detection scores for 374 concepts provided by City University of Hong Kong [7]. In order to robustly detect a concept independent of size, direction and position on the screen, researchers have prepared a large amount of training data (61,901 shots), where shots are manually annotated to note the presence or absence of the concept. Then, they constructed three SVMs based on *SIFT*, color moment and wavelet texture features. Finally, for each shot, the detection score for the concept is computed as the average of outputs of the above SVMs. In this way, detection scores for 374 concepts are assigned to all shots. That is, we use a conceptual feature which is a 374-dimensional vector consisting of detection scores for 374 concepts. Next, we will explain our PSL method which utilizes both *SIFT* and conceptual features.

Algorithm 1 summarizes our PSL method. Given a set of positive examples P for a query, our PSL method outputs a set of negative examples N as a shrunken set of unlabeled examples U , where redundant unlabeled examples are filtered. Our PSL method obtains N through a two-step process. The first step from line 1 to 5 in Algorithm 1 filters unlabeled examples dissimilar to positive examples using *SIFT* features. Unlabeled examples are iteratively filtered using SVMs built on P and U consisting of the remaining unlabeled examples. The second step (lines 6 and 7) in Algorithm 1 uses the conceptual feature to filter unlabeled examples relevant to the query from U . We first obtain a set of concepts related to the query C^* based on concept detection scores in P . We then filter unlabeled

examples which are similar to P in terms of concepts in C^* . In what follows, we describe these two steps in more detail.

Algorithm 1 An overview of our PSL method

INPUT: Set of positive examples P , set of unlabeled examples U , maximum number of iterations α , ratio between numbers of unlabeled examples and clusters β , distance threshold γ , concept selection threshold δ

OUTPUT: Set of negative examples N

FIRST STEP: Filter unlabeled examples visually dissimilar to positive examples from U

$ite_id = 0$

repeat

1. ite_id++
2. Cluster U into $(|U| / \beta)$ clusters
3. Obtain the set of representative unlabeled examples RU , each of which is closest to the cluster center
4. Build an SVM using P and RU
5. From U , filter unlabeled examples that are far from the SVM's decision boundary using γ

until $ite_id == \alpha$ OR no unlabeled example is filtered

SECOND STEP: Filter unlabeled examples relevant to the query from U

6. Get the set of concepts C^* which are related to the query using δ
7. Filter unlabeled examples which are similar to P using C^*

return $N = U$

The first step aims to filter unlabeled examples visually dissimilar to positive examples. To this end, we build an SVM on P and U where unlabeled examples considered visually dissimilar to P are those that are distant from the decision boundary of the SVM. However, using all unlabeled examples would involve a prohibitive computational cost. In addition, if a subset of unlabeled examples is randomly selected from U , unlabeled examples located in certain regions of the *SIFT* feature space may not be selected. As a result, the decision boundary of the SVM may be estimated incorrectly, and then the calculated distance between positive and unlabeled examples would be incorrect. Thus, we collect a set of *representative* unlabeled examples which characterize the distribution of all unlabeled examples. For this, we group unlabeled examples into clusters using the k -means clustering algorithm and the Euclidian distance measure. It should be noted that since various different kinds of semantic content are present in unlabeled examples, their *SIFT* features are very

diverse. Therefore, a large number of clusters are required to capture the diversity of *SIFT* features in unlabeled examples. Therefore, we use a parameter β to control the number of clusters relative to the number of unlabeled examples (line 2 in Algorithm 1). In our experiment, β is set to 10 so that when $|U|=30,000$, 3,000 clusters are obtained.

After clustering, for each cluster c , the most centrally located unlabeled example is selected as the representative example u_c :

$$u_c = \min_{u_i \in U_c} \sum_{u_j \in U_c} dist(u_i, u_j) \tag{1}$$

where U_c is the set of unlabeled examples in c , u_i and u_j are the i -th and j -th unlabeled examples in U_c , and $dist(u_i, u_j)$ represents the Euclidian distance between them. Thus, u_c is selected as the unlabeled example having the minimum sum of Euclidian distances to the other unlabeled examples in c . A set of representative unlabeled examples for all clusters is denoted as RU .

Next, we build an SVM on P and RU and check whether each unlabeled example u in U is far from the decision boundary of the SVM, based on the following criterion:

$$\left| \frac{w \bullet u - b}{\|w\|} \right| > \gamma \tag{2}$$

where the left hand side represents the distance between u and the decision boundary [2]. Specifically, $\frac{w \bullet u}{\|w\|}$ represents the projection of u onto w that is the normal vector of the hyperplane, normalized by the norm of w . The term $\frac{b}{\|w\|}$ is the offset of the hyperplane from the origin. The distance between u and the hyperplane is computed as the absolute value of the difference between $\frac{w \bullet u}{\|w\|}$ and $\frac{b}{\|w\|}$. In the SVM, since w is comprised of support vectors and their weights, the distance can be computed using only the product of u and each support vector. Thus, although a non-linear SVM projects u into a higher-dimensional feature space, the distance between u and the hyperplane can be computed by the kernel trick, where the product of u and a support vector is defined by a kernel function (see [2] for more details).

An unlabeled example u is filtered if the distance defined in Eq. 2 is larger than the threshold γ . This filtering is iterated until the number of iterations reaches the maximum number α or when no further unlabeled examples are removed from U . The resulting set U only includes unlabeled examples which are similar to P in terms of the *SIFT* feature.

The second step filters unlabeled examples relevant to the query based on their conceptual similarity to positive examples. Note that if detection scores of all 374 concepts are used, some unlabeled examples are incorrectly marked as being similar to positive examples due to their similarities to concepts unrelated to the query. Thus, as shown in line 6 in Algorithm 1, we select a subset of concepts C^* related to the query. From the entire set of 374 concepts C , C^* is obtained as follows: for each concept c , we compute the average detection score of positive examples. Then, δ concepts with the highest average detection scores are selected and included in C^* . We set δ to 10 based on a preliminary experiment.

We measure the similarity of each unlabeled example u to positive examples using concepts in C^* . If u satisfies the following criterion, u is considered relevant to the query and is filtered out of U :

$$\max_{p_i \in P} Sim(u, p_i) \geq \sigma \tag{3}$$

where p_i is a positive example in P . On the left hand side, we find the positive example that is most similar to u , where $Sim(u, p_i)$ is computed using the cosine similarity measure on

detection scores of concepts in C^* . If the similarity between u and its most similar positive example is larger than the threshold σ , we consider u to be relevant to the query. It should be noted that the cosine similarity measure is sensitive to the number of selected concepts (i.e. dimensions); this similarity tends to be smaller as the number of concepts increases. Thus, for any query, we select the same number of related concepts. In addition, when a larger number of positive examples are available, a larger number of unlabeled examples are considered to be similar to positive examples. This is because an unlabeled example only needs to be similar to one positive example, as indicated in Eq. 3. Hence, we change σ depending on the numbers of positive examples. Specifically, when the number of positive examples is less than 50, σ is set to 0.8, otherwise 0.9. Finally, unlabeled examples that remain after the above two steps are used as negative examples.

4 Experimental results

We tested our PSL method using TRECVID 2009 video data [19], consisting of 219 development videos and 619 test videos from various genres including cultural, news magazine, documentary and education programming [19]. Each video is already divided into shots using an automatic shot boundary detection method. The development and test videos consist of 36,106 and 97,150 shots, respectively. Our PSL method is evaluated on the following nine queries out of the 24 queries in TRECVID 2009 search task:

- *Query 1:* A crowd of people, outdoors, filling more than half of the frame area
- *Query 2:* A view of one or more tall buildings and the top story visible
- *Query 3:* A closeup of a hand, writing, drawing, coloring, or painting
- *Query 4:* Exactly two people sitting at a table
- *Query 5:* A street scene at night
- *Query 6:* Printed, typed, or handwritten text, filling more than half of the frame area
- *Query 7:* One or more people, each at a table or desk with a computer visible
- *Query 8:* One or more people, each sitting in a chair, talking
- *Query 9:* One or more ships or boats, in the water

We selected these queries because our PSL method constructs a classifier on an image feature (*SIFT* feature), and therefore our method should be evaluated on queries where the image feature is appropriate for retrieving relevant shots. We excluded queries where a motion or audio feature seemed to be important for retrieval, such as “a road taken from a moving vehicle through the front window” and “a person playing a piano”. In addition, for a meaningful evaluation of negative examples selected by our PSL method, it is desirable to use queries where reasonable retrieval performance can be obtained. If not, it is difficult to appropriately calculate the similarity between positive and unlabeled examples. In TRECVID 2009 search task, for some queries, such as “people shaking hands” and “shots of a microscope”, the retrieval performance is quite low even with state-of-the-art methods [13, 20, 31]. Thus, we exclude such queries from this experiment.

Considering the high-dimensionality of our shot representation (i.e. 1,000-dimensional BoVW on *SIFT* features), a sufficient number of positive examples are needed to obtain reasonable retrieval performance. However, in TRECVID 2009 search task, only a small number of positive examples are provided for each query. Thus, in addition to those positive examples, we manually selected more positive examples from development videos.

In section 4.3, we will discuss how the performance of our PSL method is varied depending on numbers of positive examples.

The retrieval for each query is conducted as follows: given positive examples, our PSL method is used to select negative examples among unlabeled examples in development videos. To filter unlabeled examples visually dissimilar to positive examples, our PSL method iteratively builds SVMs with the Radial Basis Function (RBF) kernel. In each iteration, SVM parameters are determined by conducting three-fold cross validation on the set of positive examples and a set of representative unlabeled examples (see line 4 in Algorithm 1). After completion of the PSL method, we retrieve relevant shots to the query from test videos by building an SVM on the *SIFT* feature using positive examples and selected negative examples. The parameters for this SVM are determined by three-fold cross validation on positive and selected negative examples. The final retrieval result is obtained as a ranking of shots in terms of SVM probabilistic outputs [3]. Each SVM probabilistic output represents the relevance of a shot to the query. The retrieval performance is evaluated based on ‘(inferred) Average Precision’ (AP), which is used in TRECVID 2009 search task [19]. A large AP indicates that relevant shots were ranked higher. This AP score is used to evaluate the usefulness of negative examples selected by our PSL method, where a large AP is obtained if useful negative examples are selected.

4.1 Effectiveness of negative examples selected by our PSL method

In order to examine the usefulness of negative examples selected through our PSL method, we compare the following three retrieval types, which use the same positive examples with different negative examples:

- *All*: An SVM is constructed by considering all unlabeled examples to be negative.
- *PSL*: An SVM is constructed using negative examples selected by our PSL method.
- *Random*: An SVM is constructed by considering randomly selected unlabeled examples as negative.

Different negative examples are selected in different runs of *PSL*, due to *k*-means clustering on unlabeled examples, in which different sets of representative unlabeled examples are obtained depending on the randomly selected initial cluster centers (see line 3 in Algorithm 1). Thus, *PSL* is performed 10 times for each query. Meanwhile, *Random* is conducted 10 times using different sets of randomly selected negative examples. The number of negative examples in *Random* is the same as the average number of negative examples selected in 10 runs of *PSL*.

Figure 2 shows a performance comparison between *All*, *PSL* and *Random*. The retrieval performance of *PSL* and *Random* is represented by a boxplot. Here, the bottom and top lines represent the minimum and maximum APs of the 10 retrieval results. The bottom and top of the box represent the lower and upper quartiles and the inside line represents the median. In addition, for each query, the row starting with “+” represents the number of positive examples. Rows starting with “-” represent the numbers of negative examples used in *All* and *PSL*, respectively. From Fig. 2, we can see that for all queries, the median AP score of *PSL* is higher than *All* and *Random*. In addition, on comparing the boxplots of *PSL* and *Random*, we can see that the variation of the former’s APs is smaller than the latter’s. Therefore, negative examples selected by *PSL* are very useful for accurate and stable retrieval.

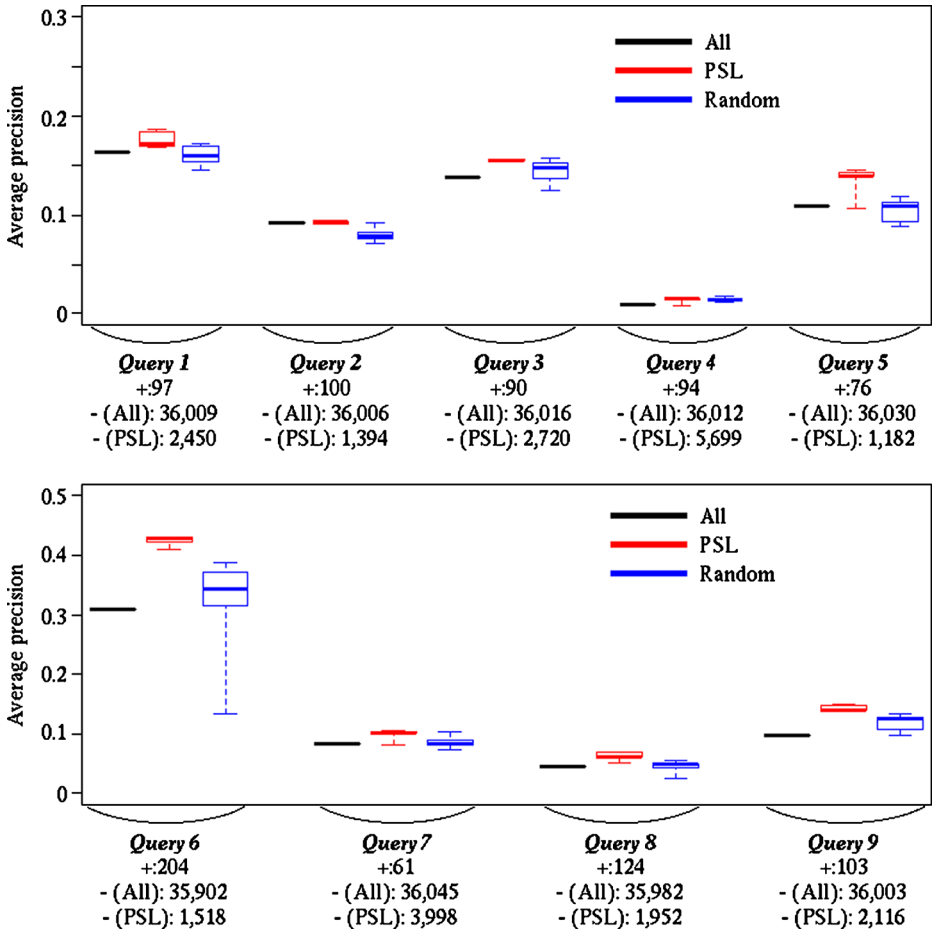


Fig. 2 Performance comparisons of *All*, *PSL* and *Random*

Figure 3 shows a comparison between *All*, *PSL* and *Random* in terms of computation time. For each query, the left, middle and right bars respectively represent computation times of *All*, *PSL* and *Random*, respectively. These computation times are

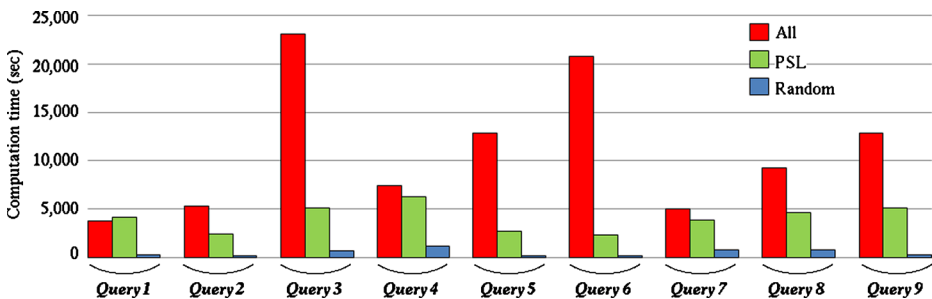


Fig. 3 Computation time comparisons between *All*, *PSL* and *Random*

required to obtain the retrieval results shown in Fig. 2. For both *PSL* and *Random*, the bar represents the average computation time of 10 retrievals. As can be seen from Fig. 3, the computation time for *PSL* is much less than the computation time for *All*. At this point, most of *PSL*'s computation time is spent on negative example selection, while almost all of the computation time for *All* is spent on SVM construction. Thus, the small number of negative examples selected by *PSL* are very important for fast retrieval, and *PSL*'s retrieval performance is higher than that of *All*. Also, the computation time for *PSL* is much higher than that for *Random*. However, the former is significantly superior to the latter in terms of retrieval performance, as shown in Fig. 2. In order to reduce the computation time for *PSL*, we plan to parallelize it using multiple processors.

4.2 Effectiveness of the hybrid use of visual and conceptual features

In order to examine the effectiveness of using both *SIFT* and conceptual features, we test the original *PSL* method (*PSL*) against three modified versions:

- *PSL_{first}*: This version only performs the first step of filtering unlabeled examples that are visually dissimilar to positive examples. This is used to evaluate the effectiveness of the second step for filtering unlabeled examples relevant to the query.

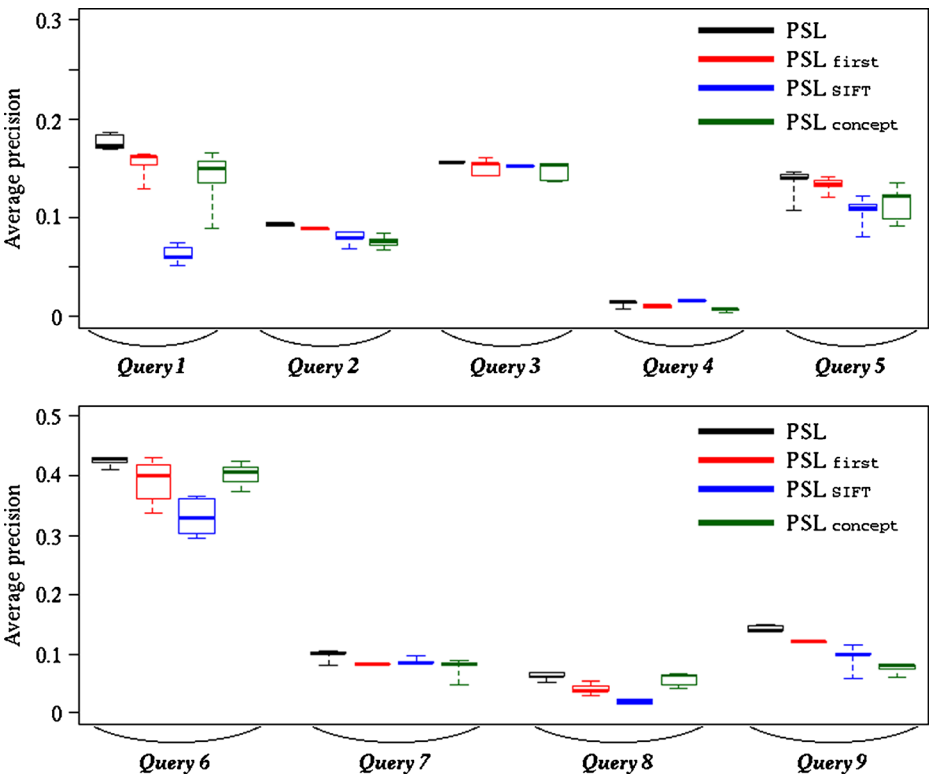


Fig. 4 Performance comparisons between *PSL*, *PSL_{first}*, *PSL_{SIFT}* and *PSL_{concept}*

- PSL_{SIFT} : This version performs both the first and second steps using only the *SIFT* feature. This is used to study the advantage of using the conceptual feature in the second step.
- $PSL_{concept}$: This version performs both the first and second steps using only the conceptual feature, and is used to study the advantage of using the *SIFT* feature in the first step.

Figure 4 shows a performance comparison between *PSL* and these three modified versions, using the same set of positive examples as in Fig. 2. In Fig. 4, except for *Query 4*, *PSL* is superior to the modified versions. This validates the effectiveness of the hybrid use of *SIFT* and conceptual features. The reason for the inferior performance of *PSL* in *Query 4* compared to the modified version PSL_{SIFT} is the lack of concepts that represent “two people” and “a table”. Hence, the *SIFT* feature is more useful than the conceptual feature for representing two people and the table. To overcome this, in section 5, we will discuss a method which can automatically and gradually increase the number of concepts.

4.3 Dependency on numbers of positive examples

In QBE, a user does not always provide as many positive examples as are shown in Fig. 2. Thus, we examine the change in retrieval performance when negative examples are collected using different numbers of positive examples. Figure 5 shows the performance of *PSL* with 10, 20, 30 and 50 positive examples (see the horizontal axis of each graph). For a query, *PSL* is run 10 times by randomly selecting a specific number of positive examples from the whole set of positive examples in Fig. 2. In Fig. 5, a diamond-shaped point represents the mean of average precisions of *PSL* using a certain number of positive examples. For comparison, we also show the change in the retrieval performance of *Random* using the same set of positive examples as *PSL* and the same number of negative examples randomly selected from unlabeled examples. In Fig. 5, a rectangular point represents the mean of average precisions of *Random* when using a specific number of positive examples.

As can be seen from Fig. 5, *PSL* outperforms *Random* except in the case of 10 positive examples in *Queries 4, 5 and 8*. That is, *PSL* can select negative examples which are more useful than *Random*, when more than 20 positive examples are available. This indicates the practical utility of *PSL* because it is not too difficult to collect more than 20 positive examples for a query using online image/video search engines. Finally, one reason why *PSL* is sometimes outperformed by *Random* with 10 positive examples is overfitting of an SVM. We will discuss this problem in detail in section 4.5.

4.4 Comparison to other partially supervised learning methods

In this section, we compare *PSL* with other partially supervised learning method. We implemented ‘Positive examples and Negative examples Labeling Heuristic’ (*PNLH*), proposed by Fung et al. [6]. Roughly speaking, *PNLH* first selects ‘reliable negative examples’ as unlabeled examples which are clearly dissimilar to positive examples. Then, the set of reliable negative examples is iteratively extended by selecting, as ‘additional negative examples’, unlabeled examples that are more similar to already selected reliable negative examples than positive examples. As a result, negative examples are selected, without selecting negative examples which are indistinguishable from positive examples.

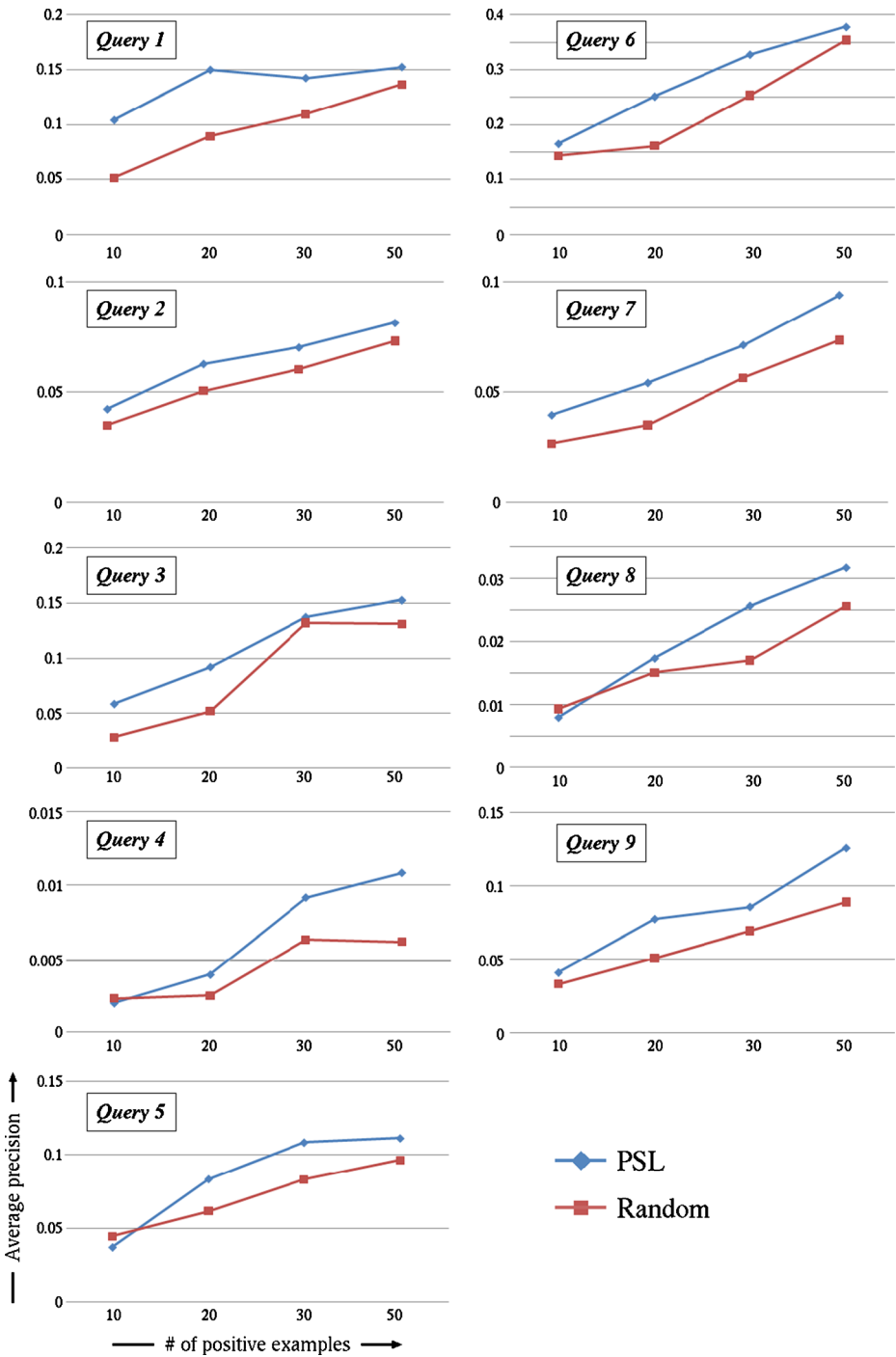


Fig. 5 Retrieval performance of PSL and Random with a varying number of positive examples

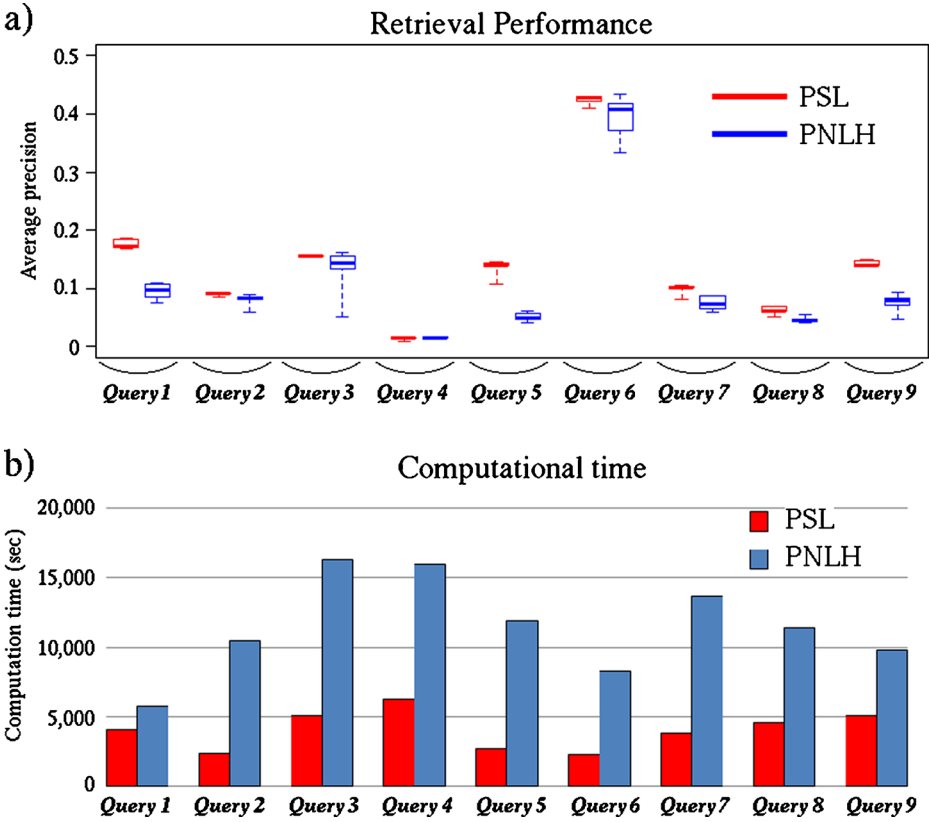


Fig. 6 Retrieval performance and computational time comparison between *PSL* and *PNLH*

Figure 6(a) shows a comparison between *PSL* and *PNLH* in terms of retrieval performance. Both of them use positive examples in Fig. 2 (*PSL*'s performance is the same as that of Fig. 2). Since *PNLH* conducts *k*-means clustering on reliable negative examples, the negative examples selected are varied depending on randomly selected cluster centers. Thus, for each query, the right boxplot represents the variation in *PNLH*'s performance over 10 retrieval results (the left boxplot represents *PSL*'s performance). As can be seen from Fig. 6(a), for *Queries 1, 2, 5, 7, 8* and *9*, the median performance of *PSL* is higher than the maximum performance of *PNLH*. In addition, for *Queries 3* and *6*, *PNLH*'s retrieval results are less accurate than *PSL*'s retrieval results. Thus, *PSL* is more effective than *PNLH* for QBE. Figure 6(b) shows a comparison between *PSL* and *PNLH* in terms of computation time. *PNLH* collects a large number of negative examples (more than 20,000) while *PSL* collects a small number of negative examples useful for accurate retrieval. Thus, *PSL*'s computation time is much shorter than that of *PNLH*.

These limitations of *PNLH* imply that it is not useful to select negative examples which are clearly distinguishable from positive examples. In related work, Yan et al. proposed a video retrieval method that selects negative examples most dissimilar to positive examples [26]. However, this method would not work effectively, similar to *PNLH*. Finally, we also implemented 'Positive Example Based Learning' (*PEBL*) proposed by Yu et al. [28]. *PEBL*

first selects reliable negative examples clearly dissimilar to positive examples. Then, additional negative examples are iteratively selected by classifying unlabeled examples with an SVM, which is constructed using positive examples and already selected negative examples. If unlabeled examples are classified as negative by the SVM, they are assumed to be additional negative examples. However, in our experiments, for all queries, *PEBL* simply selects all unlabeled examples as negative examples. One reason for this is that an SVM is constructed using positive examples, which are much less in number than negative examples. Consequently, the region of positive examples in the feature space becomes very small, and all unlabeled examples are classified as negative. To resolve this problem, *PEBL* needs to be improved so that the SVM does not classify unlabeled examples based on a binary criterion (i.e. based on whether they are located on the positive or negative side of the decision boundary). Instead, it should classify them using some continuous-valued measure, such as SVM probabilistic output.

4.5 PSL on different features and its comparison to state-of-the-art retrieval methods

Finally, we incorporate our PSL method into the retrieval method we developed in [18], and compare its retrieval performance to state-of-the-art video retrieval methods. The method in [18] addresses the fact that, even for the same query, relevant shots contain significantly different features due to variation in camera techniques and settings. To retrieve such a variety of relevant shots, we use *rough set theory*, which is a set-theoretic classification method to extract ‘rough’ descriptions of a class from imprecise or noisy data [8]. The term ‘rough’ means that rough set theory does not extract a single classification rule to characterize which characterizes the entire set of positive examples. Instead, it extracts multiple rules that characterize different subsets of positive examples. By accumulating shots retrieved by such rules, a variety of relevant shots can be retrieved.

Rough set theory requires imperfect features for classifying positive and negative examples. On the other hand, if positive and negative examples can be classified perfectly, rough set theory only extracts a small number of rules which characterize the entire set of positive examples. Such rules are not useful for extending the range of shots that can be retrieved. Thus, we construct an SVM using a subset of positive and negative examples, and use its classification result as a feature in rough set theory [18]. That is, we aim to leave the possibility of incorrectly classifying positive and negative examples which are excluded from the SVM construction. In this spirit, we build many SVMs using different subsets of positive and negative examples.

Furthermore, to retrieve a variety of relevant shots, it is important to use ‘diverse’ features that characterize different shots. To achieve this, SVMs are built using various types of features, such as *SIFT*, *Dense SIFT*, *Opponent SIFT*, *RGB SIFT*, *Hue SIFT* and *RGB Histogram* [15]. These characterize different color and edge properties in local image regions in a shot. Each feature is represented using the 1,000-dimensional BoVW representation described at the beginning of section 3. In addition, even for the same feature, when only a small number of positive examples are available, classification results of SVMs are varied significantly depending on the selected positive examples [21]. Thus, we can increase the diversity of features in rough set theory by building SVMs on different subsets of positive examples. In [18], each SVM is built by randomly selecting positive examples from all positive examples, and randomly selecting unlabeled examples as negative examples.

We incorporate our PSL method into the above rough set theory. For the construction of an SVM, our PSL method is used to select negative examples based on a subset of randomly selected positive examples. We build three such SVMs for each of the above six features. In other words, classification results of 18 SVMs are used as features in rough set theory. Rules are extracted by applying rough set theory to all positive examples and the union of negative examples selected for constructing each SVM. Finally, we retrieve shots that match many extracted rules. For simplicity, this video retrieval method is called *RST_PSL*.

We compare *RST_PSL* to methods developed for TRECVID 2009 search task (fully automatic category) [19]. In this task, for each query, TRECVID provides about 10 positive examples, which consist of example shots selected from TRECVID 2009 development videos, and example images selected from the internet. Since concept detection scores are not provided for example images, our PSL method filters unlabeled examples relevant to the query using the concept detection scores of example shots.

Many methods in TRECVID 2009 search task use a conceptual feature in the retrieval process, while *RST_PSL* uses it only for selecting negative examples and performs retrieval using visual features. Thus, we compare *RST_PSL* to methods using visual features. Table 2 shows a performance comparison between *RST_PSL* and methods developed at University of Amsterdam (*UvA*) [20] and National Institute of Informatics, Japan (*NII*) [9]. The *UvA* and *NII* methods are the first and second ranked methods using visual features. The third, fifth and sixth rows in Table 2 show the average precisions of the above methods for seven queries. We omit *Query 3* and *Query 4* because the relevant shots are considerably different to the provided positive examples, and no methods using visual features can achieve reasonable performance with these. As can be seen from the top row in Table 2, queries can be divided into two categories based on the retrieval results of *RST_PSL*. The first category consists of *Queries 1, 5 and 6* where *RST_PSL* achieves accurate retrieval. The second category consists of *Queries 2, 7, 8 and 9* where *RST_PSL* needs improvement.

As shown in Table 2, for the first category, the performance of *RST_PSL* is comparable or superior to the *UvA* and *NII* methods. However, for the second category, the performance of *RST_PSL* is comparable to *NII* method, but is inferior to *UvA*. One reason for this is ‘overfitting’ of an SVM (this is also why *PSL* is outperformed by *Random* in some cases using 10 positive examples in Fig. 5). Specifically, using negative examples similar to positive examples causes that, when only a small number of positive examples are available, the SVM is very specific to these positive examples. In other words, the region of positive examples in the visual feature space is very small. As a result, except for shots which are very similar to positive examples, the SVM assigns the same probabilistic outputs for shots, and it cannot decide whether they are relevant or irrelevant to the query.

Table 2 Retrieval performance comparisons between *RST_PSL* and methods developed at University of Amsterdam (*UvA*) and National Institute of Informatics (*NII*)

	Successful queries			Queries requiring improvement			
	<i>Query 1</i>	<i>Query 5</i>	<i>Query 6</i>	<i>Query 2</i>	<i>Query 7</i>	<i>Query 8</i>	<i>Query 9</i>
<i>RST_PSL</i>	0.115	0.075	0.31	0.041	0.028	0.014	0.019
<i>RST_PSL</i> with additional positive examples				0.096 (+10)	0.061 (+10)	0.053 (+50)	0.081 (+10)
<i>UvA</i>	0.096	0.069	0.48	0.079	0.042	0.058	0.061
<i>NII</i>	0.129	0.054	0.135	0.013	0.008	0.023	0.036

Note that overfitting is not so serious for queries in the first category, because positive examples and relevant shots are visually very similar. A small region of positive examples is enough to retrieve relevant shots.

To investigate the influence of overfitting on *RST_PSL*, we use more positive examples in addition to positive examples provided by TRECVID, and examine the improvement in retrieval performance. Additional positive examples are randomly selected from all positive examples in Fig. 2. The fourth row in Table 2 shows the retrieval performance of *RST_PSL* where the number of additional positive examples is given in parentheses. We can see that by using only ten additional positive examples, for *Queries 2, 7* and *9*, the performance of *RST_PSL* is significantly improved and outperforms *UvA*. On the other hand, for *Query 8*, 50 additional positive examples are needed to achieve a performance similar to *UvA*. Before this result, *RST_PSL* with 30 additional positive examples had an average precision 0.049 (the above 50 additional positive examples are obtained by adding 20 randomly selected positive examples to these 30 additional positive examples). It can be considered that for *Query 8*, audio features are important to characterize talking people. Visual features are insufficient where many relevant shots are visually dissimilar to a small number of available positive examples. Thus, many positive examples are needed to achieve high retrieval performance for *Query 8*. Overall, *RST_PSL* can achieve state-of-the-art retrieval performance for queries where similarities between positive examples and relevant shots are appropriately defined by visual features. In addition, about 20 positive examples are needed to avoid overfitting.

5 Conclusion and future work

In this paper, we introduced a novel PSL method which selects a small number of useful negative examples for QBE. Our PSL method performs two filtering steps using both visual (*SIFT*) and conceptual features. The first step filters unlabeled examples which are visually dissimilar to positive examples using the *SIFT* feature. We iteratively filter unlabeled examples that are far from the decision boundary of an SVM, which is built using positive examples and remaining unlabeled examples. The second step filters unlabeled examples relevant to the query. Such unlabeled examples are filtered based on their similarity to positive examples in terms of the conceptual feature. Finally, unlabeled examples that remain are regarded as negative examples, and are used to construct a retrieval model. Experimental results validate the effectiveness of the combined use of both visual and conceptual features. In addition, compared to methods using tens of thousands of negative examples, our PSL method can achieve comparable or even superior performance while requiring much less computation time. Furthermore, on comparison with methods developed for TRECVID 2009 search task, we found that video retrieval using our PSL method achieves state-of-the-art performance for some queries.

We plan to extend our PSL method as follows. While the current method only makes use of image features, we aim to use temporal features such as 3-dimensional *SIFT* [17] and acoustic features such as the Mel-Frequency Cepstrum Coefficient. The computation time required for our PSL method is much less than that needed when all unlabeled examples are considered to be negative examples. However, it is still far from being of practical use. Thus, we plan to parallelize our PSL method on multiple processors, using *MapReduce*, which is a parallel programming model for efficiently distributing/merging large-scale data based on a simple data structure [4].

Furthermore, our PSL method relies on a conceptual feature to filter unlabeled examples relevant to a query. However, as was shown by *Query 4* in Fig. 4, when there is no concept that characterizes a query, the quality of selected negative examples is worse. In this case, the visual feature is more appropriate for filtering unlabeled examples relevant to the query. To switch between the conceptual and visual features, we plan to develop a ‘query difficulty estimation’ method, which can estimate the difficulty of identifying unlabeled examples relevant to the query using a conceptual feature. If this difficulty is low, then many unlabeled examples identified using all related concepts can also be identified using each related concept [27]. Otherwise, unlabeled examples identified using all related concepts are significantly different from unlabeled examples identified using a related concept. Let us consider the query “a red car moves on a street”. We suppose that the three concepts, *Red*, *Car*, and *Street*, are related to the query. Several unlabeled examples identified by a combination of the three concepts seem to also be identified by *Car* (or *Street*). However, very few of them seem to be identified by *Red*. Thus, we can infer that the visual feature is more appropriate than the conceptual feature for filtering unlabeled examples relevant to the query.

Finally, in order to increase the number of concepts in the conceptual feature, retrieval results similar to the above, where unlabeled examples relevant to the query are filtered using the visual feature, will be used as the detection result of an additional concept. That is, SVM probabilistic outputs assigned to shots in the retrieval process are considered to be detection scores for the additional concept. The approach of using retrieval results as concept detection results has been validated, as it was used in the top performing method in TRECVID 2009 search task [31]. Therefore, when we increase the number of concepts as above, our PSL method can gradually and automatically improve to select more useful negative examples.

Acknowledgement This research is supported in part by Strategic Information and Communications R&D Promotion Programme (SCOPE) by the Ministry of Internal Affairs and Communications, Japan.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

References

1. Akbani R, Kwak S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: Proc. of ECML 2004, pp. 39–50
2. Burges C (1998) A tutorial on support vector machines for pattern recognition. *Data Min Knowl Discov* 2(2):121–167
3. Chang C, Lin C (2011) LIBSVM: a library for support vector machines. *ACM Trans Intell Syst Technol* 2(27):1–27
4. Chu C et al (2006) Map-reduce for machine learning on multicore. In: Proc. of NIPS 2006, pp. 281–288
5. Elkan C, Noto K (2008) Learning classifiers from only positive and unlabeled examples. In: Proc. of KDD 2008, pp. 213–220

6. Fung G, Yu J, Ku H, Yu P (2006) Text classification without negative examples revisited. *IEEE Trans Knowl Data Eng* 18(1):6–20
7. Jiang Y, Ngo C, Yang J (2007) Towards optimal bag-of-features for object categorization and semantic video retrieval. In: *Proc. of CIVR 2007*, pp. 494–501
8. Komorowski J, Øhrn A, Skowron A (2002) The ROSETTA rough set software system. In: Klösgen W, Zytkow J (eds) *Handbook of data mining and knowledge discovery*, chap. D.2.3, Oxford University Press
9. Le D et al (2009) National Institute of Informatics, Japan at TRECVID 2009. In: *Proc. of TRECVID 2009*, pp. 281–288
10. Li X, Snoek C (2009) Visual categorization with negative examples for free. In: *Proc. of MM 2009*, pp. 661–664
11. Liu B, Lee W, Yu P, Li X (2002) Partially supervised classification of text documents. In: *Proc. of ICML 2002*, pp. 387–394
12. Natsev A, Naphade M, Tešić J (2005) Learning the semantics of multimedia queries and concepts from a small number of examples. In: *Proc. of ACM MM 2005*, pp. 598–607
13. Ngo C et al (2009) VIREO/DVM at TRECVID 2009: High-level feature extraction, automatic video search and content-based copy detection. In: *Proc. of TRECVID2009*, pp. 415–432
14. Peng Y, Yao J (2010) AdaOUBOost: adaptive over-sampling and under-sampling to boost the concept learning in large scale imbalanced data sets. In: *Proc. of MIR 2010*, pp. 111–118
15. Sande K, Gevers T, Snoek C (2010) Evaluating color descriptors for object and scene recognition. *IEEE Trans Pattern Anal Mach Intell* 32(9):1582–1596
16. Schohn G, David C (2000) Less is more: active learning with support vector machines. In: *Proc. of ICML 2000*, pp. 839–846
17. Scovanner P, Ali S, Shah M (2007) A 3-dimensional SIFT descriptor and its application to action recognition. In: *Proc. of ACM MM 2007*, pp. 357–360
18. Shirahama K, Matsuoka Y, Uehara K (2011) Video event retrieval from a small number of examples using rough set theory. In: *Proc. of MMM 2011*, pp. 96–106
19. Smeaton A, Over P, Kraaij W (2006) Evaluation campaigns and TRECVID, In: *Proc. of MIR 2006*, pp. 321–330
20. Snoek C et al (2009) The MediaMill TRECVID 2009 semantic video search engine. In: *Proc. of TRECVID2009*, pp. 226–238
21. Tao D, Tang X, Li X, Wu X (2007) Asymmetric bagging and random subspace for support vector machine-based relevance feedback in image retrieval. *IEEE Trans Pattern Anal Mach Intell* 28(7):1088–1099
22. Tax D, Duin R (2001) Uniform object generation for optimizing one-class classifiers. *J Mach Learn Res* 2:155–173
23. Tešić J, Natsev A, Smith J (2007) Cluster-based data modeling for semantic video search. In: *Proc. of CIVR 2007*, pp. 595–602
24. Tsang I, Kwok J, Cheung P (2005) Core vector machines: fast SVM training on very large data sets. *J Mach Learn Res* 6:363–392
25. Vapnik V (1998) *Statistical learning theory*. Wiley-Interscience
26. Yan R, Hauptmann A, Jin R (2003) Negative pseudo-relevance feedback in content-based video retrieval. In: *Proc. of ACM MM 2003*, pp. 343–346
27. Yom-Tov E, Fine S, Carmel D, Darlow A (2005) Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In: *Proc. of SIGIR 2005*, pp. 512–519
28. Yu H, Han J, Chang K (2004) PEBL: web page classification without negative examples. *IEEE Trans Knowl Data Eng* 16(1):70–81
29. Yuan J, Li J, Zhang B (2006) Learning concepts from large scale imbalanced data sets using support cluster machines. In: *Proc. of ACM MM 2006*, pp. 441–450
30. Zhang J, Marszałek M, Lazebnik S, Schmid C (2007) Local features and kernels for classification of texture and object categories: a comprehensive study. *Int J Comput Vis* 73(2):213–238
31. Zhao Z et al (2009) BUPT-MCPRL at TRECVID 2009. In: *Proc. of TRECVID 2009*, pp. 42–53



Kimiaki Shirahama received his B.E., M.E. and D.E. degrees in Engineering from Kobe University, Japan in 2003, 2005 and 2011, respectively. Currently, he is an assistant professor in the Graduate School of Economics at Kobe University. His research interests include multimedia data processing, data mining and virtual reality. He is a member of ACM SIGKDD, ACM SIGMM, the Institute of Image Information and Television Engineers in Japan (ITE), Information Processing Society of Japan (IPSJ) and the Institute of Electronics, Information and Communication Engineering in Japan (IEICE).



Yuta Matusoka received his B.E. M.E. degrees in Engineering from Kobe University, Japan in 2009 and 2011, respectively. He joined NTT Data Kansai Corporation as a system engineer in April 2011. In Kobe University, he studied about image/video data retrieval and machine learning.



Kuniaki Uehara received his B.E., M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan in 1978, 1980 and 1984, respectively. He was an assistant professor in the Institute of Scientific and Industrial Research, Osaka University from 1984 to 1990, an associate professor in the Department of Computer Science and Systems Engineering, Kobe University from 1990 to 1997, a professor in the Research Center for Urban Safety and Security, Kobe University from 1997 to 2002 and a professor in the Graduate School of Science and Technology, Kobe University from 2002 to 2009. Currently, he is a professor in the Graduate School of System Informatics, Kobe University. From 1989 to 1990, he was also a visiting assistant professor at Oregon State University. He has published a number of books, articles and conference papers in wide areas of artificial intelligence, especially in machine learning, natural language processing and intelligent software engineering. He is a member of AAAI, Information Processing Society of Japan (IPSJ), the Japan Society for Artificial Intelligence (JSAI), the Institute of Electronics, Information and Communication Engineering in Japan (IEICE), the Mathematical Linguistic Society of Japan and Japan Society for Software Science and Technology (JSSST).