

Event retrieval in video archives using rough set theory and partially supervised learning

Kimiaki Shirahama · Yuta Matsuoka · Kuniaki Uehara

Published online: 20 January 2011

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

Abstract This paper develops a query-by-example method for retrieving shots of an event (event shots) using example shots provided by a user. The following three problems are mainly addressed. Firstly, event shots cannot be retrieved using a single model as they contain significantly different features due to varied camera techniques, settings and so forth. This is overcome by using rough set theory to extract multiple classification rules with each rule specialized to retrieve a portion of event shots. Secondly, since a user can only provide a small number of example shots, the amount of event shots retrieved by extracted rules is inevitably limited. We thus incorporate bagging and the random subspace method. Classifiers characterize significantly different event shots depending on example shots and feature dimensions. However, this can result in the potential retrieval of many unnecessary shots. Rough set theory is used to combine classifiers into rules which provide greater retrieval accuracy. Lastly, counter example shots, which are a necessity for rough set theory, are not provided by the user. Hence, a partially supervised learning method is used to collect these from shots other than example shots. Counter example shots, which are as similar to example shots as possible, are collected because they are useful

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

for characterizing the boundary between event shots and the remaining shots. The proposed method is tested on TRECVID 2009 video data.

Keywords Query by example · Rough set theory · Bagging · Random subspace · Small sample size problem · Partially supervised learning

1 Introduction

In recent times, a large number of videos have been stacked as video archives. Viewing these is a time consuming process since videos are essentially temporal media. Thus, there is great demand to develop a method which can efficiently retrieve interesting shots (or video segments) from the archive. Users are usually interested in retrieving shots which match semantic content rather than physical content such as RGB pixel values. In this paper, the term *event* is used to denote a query regarding semantic content. The shots matching the event are termed *event shots* while *non-event shots* refer to the remaining shots. The retrieval of event shots is termed *event retrieval*.

The proper representation of an event is very important. Existing approaches can be roughly classified into two types, namely, the *Query-By-Keyword* (QBK) and *Query-By-Example* (QBE) approaches. With QBK, a user represents an event by using keywords and shots are subsequently retrieved by matching with defined keywords. With QBE, a user provides example shots to represent an event and shots are then retrieved based on their similarity to example shots in terms of features. For example, consider the event ‘people appear with computers’ depicted in Fig. 1. Assume that an event shot, *Shot 1*, is annotated with the words ‘people’ and ‘computer’. To retrieve *Shot 1* using QBK, a user needs to enter the keywords ‘people’ and ‘computer’. However, the keywords ‘programmer’ or ‘internet user’ might be entered instead, which would not match the annotated words for *Shot 1*, despite matching the actual event. This ambiguity relating to semantic content makes it difficult for the user to appropriately represent events using keywords. Alternatively, with QBE, the event is defined using features contained in example shots, such as *Ex. 1* in Fig. 1. This eliminates the ambiguity associated with semantic content found in the QBK approach.

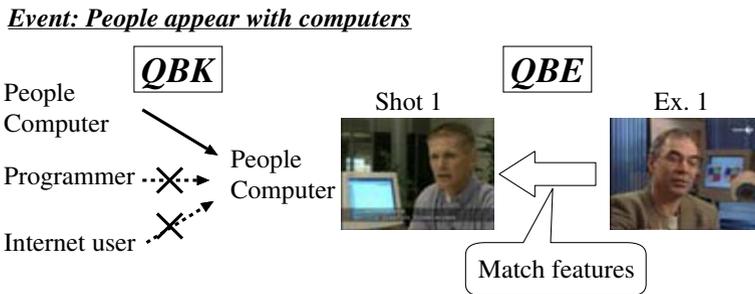


Fig. 1 Comparison between the QBK and QBE approaches for the event ‘people appear with computers’

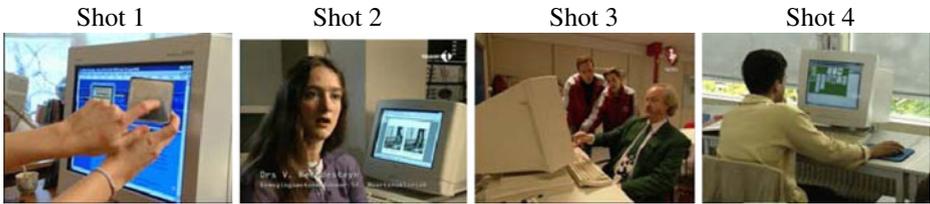


Fig. 2 An example of a variety of event shots for the event ‘people appear with computers’

The QBE approach offers the added advantage of not requiring predefined retrieval models. In traditional QBK methods, a retrieval model needs to be prepared for each event [15, 30], and recent QBK methods prepare classifiers for assessing the relevance of keywords¹ defining each event [22, 31]. However, it is impractical to prepare retrieval models and classifiers for all possible events. In comparison, a retrieval model is constructed on the fly from example shots in the QBE approach. In other words, as long as example shots are provided, QBE can perform retrieval for any event. In this paper, we describe the development of a QBE method.

The following three problems in the QBE method have been addressed.

1. Large variety of event shots Event shots are taken using different camera techniques and settings. For example, in Fig. 2, *Shot 1* depicts the user’s hands with the computer monitor in a tight shot. *Shot 2* shows the face of a person with the front of the computer monitor while *Shot 3* shows a computer monitor from the side. *Shot 4* captures the back of a person facing the computer screen. This illustrates that objects related to the same event can be depicted in several ways. Furthermore, event shots show numerous objects unrelated to the event. For example, among four shots in Fig. 2, the background and peoples’ clothing are different and a caption is visible in *Shot 2*, but not in any other shots. Thus, even for the same event, event shots not only contain significantly different features, but also many redundant features. Therefore, a single retrieval model is not capable of retrieving a large variety of event shots.

Rough set theory (RST) is a set-theoretic classification method for extracting rough descriptions of a class from imprecise (or noisy) data, and is used for retrieving a variety of event shots [13]. RST can be used to extract multiple classification rules, which can correctly identify different subsets of example shots. In other words, each classification rule is specialized for retrieving event shots characterized by certain features. Hence, by accumulating event shots retrieved with various classification rules, we can retrieve a variety of event shots.

2. Small sample size In QBE, a user can only provide a small number of example shots for an event. Since QBE, by definition, retrieves shots similar to example shots, a small number of example shots will inevitably lead to a small range of event shots. We use *bagging* and the *random subspace method* to overcome this problem.

¹These keywords are frequently called *concepts*. However, some readers may confuse them with concepts which are hierarchically organized in an ontology. In light of this, we do not use the term *concept*.

Specifically, various classifiers are built using randomly selected example shots and feature dimensions. When only a small number of example shots are available, classifiers output significantly different classification results depending on example shots [32]. In addition, classification results differ depending on feature dimensions [8]. Thus, by building various classifiers using bagging and the random subspace method, we can extend the range of event shots that can be retrieved. However, this also results in many non-event shots potentially being retrieved. To overcome this, RST is used to extract classification rules as combinations of classifiers, which can accurately retrieve event shots.

3. Lack of negative examples RST extracts classification rules to enable the discrimination of event shots from non-event shots. This requires two types of example shots, *positive examples* (p-examples), provided by a user to serve as representatives of event shots, and *negative examples* (n-examples), which are not provided by the user and serve as representatives of non-event shots. To solve the lack of n-examples, we formulate the QBE approach as *partially supervised learning* (PSL). Classification rules are extracted using p-examples and *unlabeled examples* (u-examples), which refer to shots other than p-examples. N-examples are collected from these u-examples. It can be considered that n-examples similar to p-examples are informative, because they allow the characterization of the boundary between event and non-event shots. To collect such n-examples, we devise a PSL method based on the coarse-to-fine approach. Firstly, all u-examples are regarded as n-examples, because the number of event shots included in u-examples is usually very small. Subsequently, n-examples which are dissimilar to p-examples, are iteratively filtered using a classifier built on p-examples and the remaining n-examples.

Our proposed QBE method is summarized in Fig. 3. It should be noted that our main research objective is to develop a method which can retrieve a variety

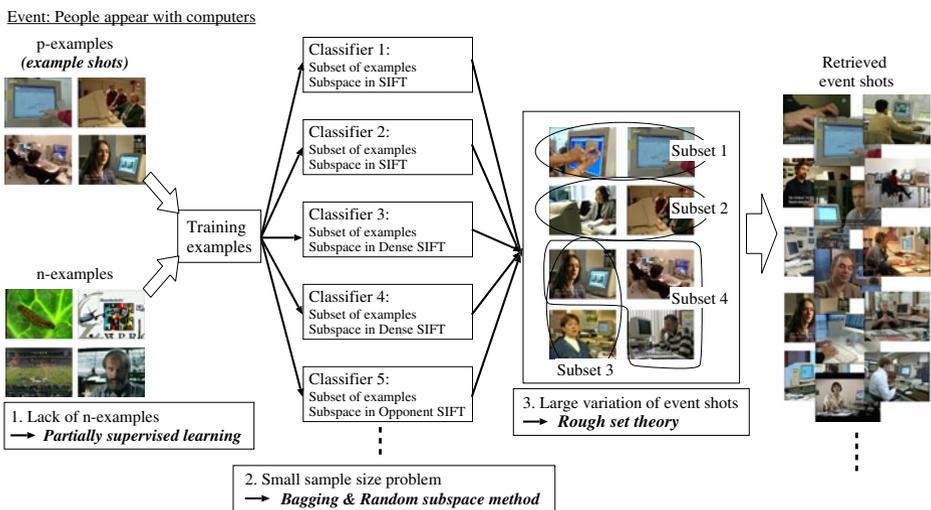


Fig. 3 An overview of our event retrieval method based on the QBE approach

of event shots only by using a small number of p-examples. Given p-examples, n-examples are collected using the PSL method. Subsequently, various classifiers are built based on bagging and the random subspace method. Lastly, RST is used to extract combinations of classifiers as classification rules, and shots matching many rules are retrieved.

2 Related works and the innovation of our research

2.1 Rough set theory

Firstly, RST determines the indiscernibility relation, which relates to whether a p-example and n-example can be discerned with respect to available features. Thereafter, multiple classification rules are extracted by combining indiscernibility relations among examples based on set theory. RST can extract classification rules without any assumption or parameter as long as indiscernibility relations can be defined.

Although methods other than RST can be used to retrieve a variety of event shots, they have inappropriate limitations. For example, a Gaussian Mixture Model (GMM) can extract multiple feature distributions of event shots [34]. However, such distributions cannot be appropriately estimated only from a small number of p-examples. In addition, without any a priori knowledge, the number of Gaussian distributions in the GMM cannot be effectively determined. The Genetic Algorithm (GA) can be used to extract multiple classification rules [7]. Each rule is encoded as a bit string (chromosome), where one bit indicates whether or not a feature is utilized. The GA combines bit strings based on the principles of biological evolution, such as crossover and mutation, to extract accurate rules. However, with no a priori knowledge, parameters in the GA, such as the number of bit strings, the probability of crossover and the probability of mutation, cannot be effectively determined.

Decision tree learning methods extract multiple classification rules in a tree-based approach [2]. Each rule is represented as a path in a tree, where p-examples and n-examples are recursively classified using a feature associated with each node. Sequential covering methods extract multiple rules in a sequential approach [2]. Each rule is sequentially extracted from p-examples, which are not characterized (covered) by already extracted rules. But, the tree-based and sequential approaches only extract one classification rule for each p-example. As a result, the number of extracted rules is inevitably small, which is insufficient for retrieving a variety of event shots. In comparison to the above methods, without any parameter setting, RST can extract various rules as minimal sets of features, which can correctly identifying different subsets of p-examples.

Traditional RST can only deal with categorical features where the indiscernibility relation between two examples can be easily defined according to whether they have the same value. In contrast, in our case, examples are represented by non-categorical and high-dimensional features. For instance, the bag-of-visual-words representation involves thousands of dimensions, each of which indicates the frequency of a local edge shape (visual word). Thus, when applying RST to QBE, the most important issue becomes the definition of indiscernibility relations among examples, that is,

the categorization of non-categorical high-dimensional features. With respect to this issue, existing approaches can be classified into the following three types:

- 1. Clustering-based** This approach groups examples into a small number of clusters. The indiscernibility relation between two examples is then defined by examining whether or not they belong to the same cluster [5, 20].
- 2. Similarity-based** This approach does not categorize a feature, but rather defines the indiscernibility relation between two examples by measuring their similarity for the feature [28, 36].
- 3. Classifier-based** This approach builds a classifier on a feature, and defines the indiscernibility relation by examining whether or not two examples are classified into the same class [25].

In our research, we have previously developed the clustering-based and similarity-based RSTs, however, they had performance limitations. In [20], we developed a clustering-based RST using k-means clustering and tested it on TRECVID 2008 video data. TRECVID is an annual international competition where a large video data is used to benchmark state-of-the-art video analysis methods [29]. However, for most of events, inferred average precisions of the clustering-based RST were nearly equal to zero. One main reason for this was the coarseness in categorizing a feature into a small number of clusters, which led to semantically different shots frequently being included in the same cluster.

We also developed a similarity-based RST in [28]. Although its performance was better than that of the clustering-based RST, it was far from the satisfactory. Table 1 provides a performance comparison between the similarity-based RST and classifier-based RST. We use the same examples, and the same features as will be described in Section 3. In the similarity-based RST, cosine similarity is used as a similarity measure, while the classifier-based RST categorizes features using Support Vector Machines (SVMs) with the Radial Basis Function (RBF) kernel. For each event, the retrieval performance is evaluated as the number of event shots within 1,000 retrieved shots. As seen in Table 1, the similarity-based RST is significantly outperformed by the classifier-based RST. The reason for the low performance of similarity-based RST is that it is difficult to appropriately measure similarities among examples for high-dimensional features.

Table 1 demonstrates the effectiveness of the classifier-based RST. In the following paragraphs, we discuss which classifiers are effective for the QBE approach, where high-dimensional features have to be categorized using only a small number of examples. The existing classifier-based RST uses different types of classifiers, such as the decision tree, nearest neighbor, naive Bayes and maximum entropy [25]. However, they are ineffective for the following two reasons. Firstly, a nearest neighbor is ineffective because the result of the similarity-based RST in Table 1 implies

Table 1 Comparison of the similarity-based RST and classifier-based RST

Event	Tall building	Flame	Computer	Helicopter/plane	Talking
Similarity-based RST	46	14	54	17	38
Classifier-based RST	144	100	150	34	67

that similarity measures do not work well for high-dimensional features. Secondly, the other classifiers rely on probabilistic distributions, such as information gains in decision tree, conditional probabilities in naive Bayes, and entropy models in maximum entropy. These are ineffective because probabilistic distributions estimated from a small number of examples tend to deviate from the true distributions [6].

In contrast, SVMs are effective when only a small number of examples are available as the margin maximization does not require any probability estimation [6]. Additionally, Vapnik [33] theorized that if the number of feature dimensions is large, the generalization error of an SVM is defined by the margin size and properties of examples, such as the diameter of the sphere enclosing examples and the number of examples. That is, from the theoretical perspective, the generalization error of the SVM is independent of the number of feature dimensions if this number is sufficiently large. Furthermore, as examples are generally not linearly separable, a kernel function is used to transform a high-dimensional feature into a higher-dimensional feature, with the above theory allowing a well generalized SVM to be built independent of the number of feature dimensions. Thus, we develop a classifier-based RST using SVMs, specifically using SVMs with the RBF kernel as it is known to be the most general kernel [9].

2.2 The problem of small sample size

As described in Section 1, bagging and the random subspace method are useful for extending the range of event shots that can be retrieved. They are additionally useful for alleviating two important problems related to small sample sizes. The first is the *class imbalance problem*, which refers to the imbalance between the number of p-examples and n-examples, significantly degrading the classification performance [1]. In our technique, numerous n-examples can be collected using the partially supervised learning method. However, the SVM built using a small number of p-examples and all collected n-examples cannot appropriately classify shots into event and non-event shots. To address this, we use bagging which combines classifiers built on different sets of randomly selected examples [3]. In other words, for a small number of p-examples, an appropriate number of n-examples are randomly selected to build an SVM. However, due to the insufficiency of n-examples, the SVM wrongly classifies many non-event shots as event shots. Thus, we combine SVMs built on different sets of randomly selected n-examples in order to consider a variety of n-examples.

The second problem is *overfitting* that a classifier can perfectly classify p-examples and n-examples, but cannot appropriately classify unseen examples. Generally, as the number of feature dimensions increases, the number of examples required to construct a well generalized classifier exponentially increases [10]. This is due to the fact that a class needs to be determined for each combination of values along different dimensions. In our case, we can only use a small number of examples (at most, a hundred of p- and n-examples in total). On the other hand, based on the bag-of-visual-words model, we represent each example as a vector with more than 1,000 dimensions. As a result, the SVM is overfit to feature dimensions which are specific to p-examples (or n-examples), but are ineffective for characterizing event shots (or non-event shots). Thus, we use the random subspace method, which combines classifiers built on randomly selected feature dimensions [8]. The original

high-dimensional feature is transformed into a lower-dimensional feature, which alleviates building an overfit SVM. By combining such SVMs, a large number of feature dimensions can be considered.

Numerous methods have been proposed to overcome the class imbalance and overfitting. For the class imbalance problem, Japkowicz [11] tested various sampling approaches, such as over-sampling of examples belonging to the minority class, under-sampling of examples belonging to the majority class, and sampling based on a classifier for examining the association of an example to the minority (or majority) class. Akbani et al. [1] used the Synthetic Minority Oversampling Technique (SMOTE), which synthetically generates new examples for the minority class based on their similarities with existing examples. Liu [17] presented various feature dimension reduction methods to overcome the problem of overfitting. For example, one method selects feature dimensions by measuring their relative importance based on an ensemble of decision trees, while a second method selects feature dimensions which are both maximally relevant and minimally redundant based on the mutual information between examples and classes. Guo and Dyer [6] proposed a method which simultaneously achieves dimension reduction and margin maximization in classifier training. As the above methods only select the best subset among examples or feature dimensions, they are not useful for extending the range of event shots that can be retrieved, unlike bagging and the random subspace method.

The application of bagging and the random subspace method in this paper is crucially different to that in the previous research [32]. In particular, Tao et al. [32] performed simple majority voting using SVMs built by bagging and the random subspace method. We refer to this majority voting as *Simple_MV*. On the other hand, our method involves majority voting using classification rules, which are extracted as combinations of SVMs by RST. We refer to this as *RST_MV*. Table 2 shows a performance comparison between *Simple_MV* and *RST_MV*. For each event, 10 retrieval results are obtained using *Simple_MV* or *RST_MV*. To perform an appropriate comparison, retrieval results are obtained by using the same set of 60 SVMs for both *Simple_MV* and *RST_MV*. The second and fourth rows represent average numbers of event shots retrieved by *Simple_MV* and *RST_MV*, respectively. These rows demonstrate that *RST_MV* outperforms *Simple_MV* for all events. A key reason for this is the difference between SVMs and classification rules. The third and fifth rows represent average numbers of event shots which are correctly classified by SVMs and classification rules, respectively, showing that classification rules are

Table 2 Performance of SVM combination by majority voting and RST

	Event	Tall building	Flame	Computer	Helicopter/plane	Talking
Majority voting	Retrieval performance	136.0	122.6	160.5	36.1	175.6
	SVM performance	84.3	67.4	90.4	22.6	98.4
RST	Retrieval performance	140.6	145.3	164.3	41.5	189.9
	Rule performance	109.6	98.6	118.8	32.4	139.5

much more accurate than SVMs. This allows *RST_MV* to achieve more accurate event retrieval than *Simple_MV*.

2.3 Negative example selection

Traditional QBE methods only use p-examples and retrieve shots similar to them [12, 23]. But, only considering similarities to p-examples cannot yield accurate event retrieval. For example, consider the event ‘a car moves’. If a p-example showing a moving red car is provided, it may seem more similar to a shot where a person wears a red piece of clothing than a shot showing a moving white car. Compared to this, if a QBE method uses n-examples and compares them to the p-example, it can be found that the edge feature characterizing a car shape is important while the color feature is not. Thus, n-examples are required to construct a retrieval model (in our case, classification rules), which characterizes features specific to p-examples.

Since event shots form only a small portion of all shots, several methods use an approach which considers randomly selected shots as n-examples [21, 31]. However, this approach is associated with the following problem. The random selection of n-examples does not consider the type of n-examples that are required to construct an accurate retrieval model. For example, if all of collected n-examples are significantly dissimilar to p-examples, we cannot identify features that are relevant to p-examples because p-examples and n-examples can be classified by using any features.

Numerous partially supervised learning (PSL) methods have been proposed to build a classifier using p-examples and unlabeled examples (u-examples). Most the existing methods adopt the two-step approach, wherein n-examples are first collected from u-examples and a classifier is subsequently built on p-examples and collected n-examples [4, 16, 37]. In such a PSL approach, one of main research issues is how to collect n-examples from u-examples. For instance, Liu et al. [16] proposed a method which selects some p-examples as ‘spy’ examples and adds them to u-examples. A naive Bayesian classifier is then built using p-examples and u-examples, where spy examples are used to set the probabilistic threshold for evaluating whether or not u-examples can be regarded as negative. Yu et al. [37] proposed a method which iteratively collects n-examples. In each iteration, the method builds an SVM from p-examples and already selected n-examples, and subsequently selects n-examples as u-examples, which are classified as negative by the SVM. Fung et al. [4] proposed an iterative method where u-examples which are more similar to already selected n-examples rather than p-examples are selected as n-examples.

However, most of existing PSL methods only collect a large number of n-examples. Due to the class imbalance problem, the use of all collected n-examples does not guarantee the construction of an accurate classifier. To overcome this, we propose a new PSL method which can select a small number of n-examples useful for building an accurate classifier like SVM. Our method is inspired by the method proposed in [38]. It was originally developed to solve the class imbalance problem using under-sampling examples of the majority class, i.e., n-examples. Yuan et al. [38] contends that since n-examples similar to p-examples characterize the class boundary, they are useful for building an accurate SVM. N-examples are iteratively filtered to collect those n-examples that are similar to p-examples. In each iteration, the method filters out n-examples that are distant from the decision boundary of the

SVM, which is built on p -examples and the remaining n -examples. As a result, the method selects a small number of n -examples that are as similar to p -examples as possible. We apply this method to PSL by regarding all u -examples as n -examples, because almost all of u -examples are non-event shots.

3 Event retrieval method

In this section, the proposed QBE method is described. We assume that the representative semantic content in each shot is shown in the middle video frame, called *keyframe*. The following six types of features are extracted from the keyframe in each shot using the color descriptor software [26]: 1. *SIFT*, 2. *Opponent SIFT*, 3. *RGB SIFT*, 4. *Hue SIFT*, 5. *Color histogram* and 6. *Dense SIFT*. The first three types of SIFT features are defined in different color spaces and have different invariance properties for lighting conditions. By using these three SIFT features, we aim to characterize local shapes of objects (e.g. corners of buildings, vehicles, human eyes etc.), regardless of changes in illumination. Since they only compute derivatives in color spaces, they do not consider absolute color values which are effective for characterizing certain objects, such as faces, flames, vegetation and so on. So, we use *Hue SIFT* and *Color histogram*. *Hue SIFT* represents a SIFT feature concatenated with a Hue histogram in HSV color space, and *Color histogram* represents a pure RGB color histogram. All of the above five types of features are extracted at interest points computed by Harris-Laplace detector, where pixel values largely change in multiple directions. But, small changes of pixel values are critical when considering certain objects such as sky, water, roads and so on. In addition, many interest points in the background may be missed due to low contrast. Thus, *Dense SIFT* is used, wherein SIFT features are extracted at interest points sampled with a fixed interval. In this way, interest points missed by Harris-Laplace detector can be compensated. Due to space limitations, the above features are not discussed any further and can be found in [26] in greater detail.

The above six types of features are represented using the bag-of-visual-words representation. For each type of feature, 1,000 visual words are extracted by clustering features at 200,000 interest points, sampled from keyframes in TRECVID 2009 development videos (219 videos) [29]. As depicted in Fig. 4, a shot is represented as a 6,000-dimensional vector where each type of feature is represented as a 1,000-dimensional vector. Based on this high-dimensional representation, we perform rough set theory extended by bagging and the random subspace method, and partially supervised learning. It should be noted that for the random subspace method, it is unreasonable to build an SVM using dimensions randomly sampled across different types of features. Thus, we build one SVM by randomly selecting dimensions in the same type of feature.

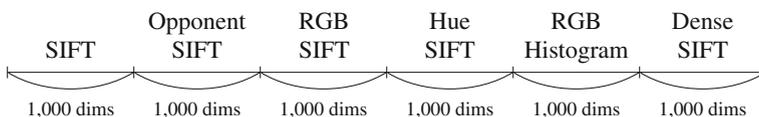


Fig. 4 The representation of a shot as a 6,000-dimensional vector

3.1 Rough set theory extended by bagging and the random subspace method

Given p-examples and n-examples, RST is used to extract classification rules, known as *decision rules*, for distinguishing event shots from non-event shots. Firstly, various SVMs are built using bagging and the random subspace method. Using each of these SVMs, the indiscernibility relation between a p-example and n-example is defined by determining whether or not the p-example and n-example are classified into the same class. Lastly, by combining such indiscernibility relations among examples, decision rules, which can discriminate subsets of p-examples from the entire set of n-examples, are extracted. The proposed RST is explained in detail in the following paragraphs.

First of all, the number of p-examples and that of n-examples are too small to characterize the distribution of event shots and that of non-event shots in a high-dimensional feature space, respectively. So, the decision boundary of an SVM tends to be inaccurate. Generally, an SVM determines the class of an example based on the binary criterion, i.e., whether the example is located on the positive or negative side of the decision boundary. However, this classification is erroneous since the decision boundary is inaccurate. To overcome this, a continuous-valued criterion is employed. Specifically, the *probabilistic output* of the SVM, which approximates the distance between an example and the decision boundary using a sigmoid function, is used [18]. Based on this, the class of an example is determined as follows. Firstly, examples are ranked in descending order of probabilistic outputs of the SVM. If an example is ranked within the top M positions, where M is the number of p-examples, its class is determined as positive, or otherwise as negative. Thus, although the decision boundary is inaccurate, examples can be robustly classified.

Classification results of SVMs can be summarized in a *decision table*, as shown in Fig. 5. Each row represents the i -th p-example, p_i ($1 \leq i \leq M$) or j -th n-example,

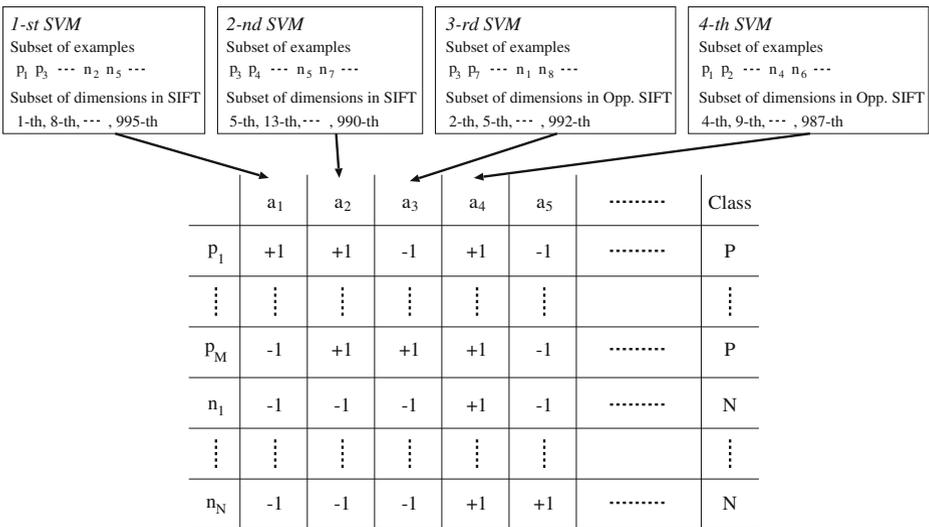


Fig. 5 Example of a decision table formed by applying RST extended by bagging and the random subspace method

n_j ($1 \leq j \leq N$). Each column a_k ($1 \leq k \leq K$) represents the classification result of the k -th SVM, where $+1$ indicates that an example is classified as positive and -1 indicates that it is classified as negative. The classification result of the k -th SVM for p_i and n_j are represented by $a_k(p_i)$ and $a_k(n_j)$, respectively. That is, each classification result can be regarded as one feature in RST. Lastly, the rightmost column indicates whether an example is positive (P) or negative (N).

Before discussing the proposed decision rule extraction method, a conceptual explanation in relation to decision rules is provided using Fig. 6. In this figure, one p-example p_1 and two n-examples n_1 and n_2 are given for the event ‘tall buildings are shown’. Let a_{k_1} and a_{k_2} represent the classification of the k_1 -th SVM built on *SIFT* feature and that of the k_2 -th SVM built on *Hue SIFT* feature, respectively. Since *SIFT* feature of p_1 is similar to the one of n_1 , the k_1 -th SVM incorrectly classifies both as positive. On the other hand, since *Hue SIFT* feature of p_1 is dissimilar to that of n_1 , the k_2 -th SVM correctly classifies p_1 and n_1 as positive and negative, respectively. The dissimilarity between *SIFT* features of p_1 's and n_2 's enables them to be correctly classified by the k_1 -th SVM. On the other hand, p_1 and n_2 cannot be correctly classified by the k_2 -th SVM due to their similar *Hue SIFT* features. Thus, in order to discriminate p_1 from n_1 and n_2 , a decision rule consisting of a_{k_1} and a_{k_2} is required. This rule indicates the combination of the k_1 -th and k_2 -th SVMs. In the following paragraphs, the extraction of such decision rules based on a logical operation is described.

In order to extract decision rules, for each pair of p_i and n_j , we first determine *discriminative features* which are useful for discriminating them. Specifically, the following set of discriminative features $f_{i,j}$ are extracted between p_i and n_j .

$$f_{i,j} = \{a_k | a_k(p_i) = +1 \wedge a_k(p_i) \neq a_k(n_j)\} \tag{1}$$

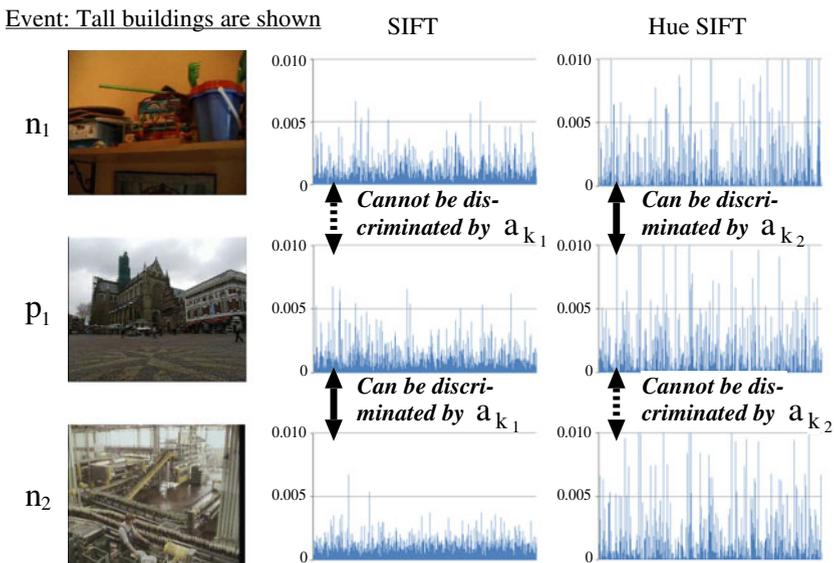


Fig. 6 The concept of decision rules extracted by RST

This signifies that a_k is included in $f_{i,j}$, if the k -th SVM correctly classifies p_i and n_j as positive and negative, respectively. In other words, p_i can be discriminated from n_j when at least one feature in $f_{i,j}$ is utilized.

Next, in order to discriminate p_i from all n -examples, the discriminative features of p_i 's are combined. This is achieved by using at least one discriminative feature in $f_{i,j}$ for all n -examples. The *discernibility function* df_i , which takes a conjunction of $\vee f_{i,j}$, is computed as follows:

$$df_i = \wedge \{ \vee f_{i,j} \mid 1 \leq j \leq N \} \tag{2}$$

Let us consider the discernibility function df_1 for one p -example p_1 and two n -examples n_1 and n_2 . Let the sets of discriminative features between p_1 and n_1 and between p_1 and n_2 be $f_{1,1} = \{a_1, a_3, a_5\}$ and $f_{1,2} = \{a_1, a_2\}$, respectively. Under this condition, df_1 can be computed as $(a_1 \vee a_3 \vee a_5) \wedge (a_1 \vee a_2)$. This is simplified as $df_1^* = (a_1) \vee (a_2 \wedge a_3) \vee (a_2 \wedge a_5)$.² Thus, p_1 can be distinguished from n_1 and n_2 , by using a_1 , the set of a_2 and a_3 , or the set of a_2 and a_5 . Similarly, each conjunction term in df_i^* represents a *reduct* which is the minimal set of features required to discriminate p_i from all n -examples.

From each reduct, we construct a decision rule in the form of an *IF-THEN* rule. A reduct r , consisting of L features, can be represented as follows.

$$r = a_{1^*} \wedge a_{2^*} \wedge \dots \wedge a_{L^*} \tag{3}$$

where a_{l^*} ($1 \leq l^* \leq L$) denotes a single feature among the total K features a_1, \dots, a_K . Recall that in (1), a_k is selected as a discriminative feature only when $a_k(p_i) = +1$. Thus, the decision rule, *rule*, constructed from r has a conditional part, where $a_{l^*}(x)$ has to be $+1$ for a shot (i.e. unseen example) x . It is represented as follows.

$$rule : IF (a_{1^*}(x) = +1) \wedge \dots \wedge (a_{L^*}(x) = +1), THEN Class = P \tag{4}$$

For example, from the reduct $(a_2 \wedge a_3)$, we can construct the decision rule *IF* $(a_2(x) = +1) \wedge (a_3(x) = +1)$, *THEN* $Class = P$. This rule indicates that if both the 2-nd and 3-rd SVMs classify x as positive, then x is an event shot. Like this, a decision rule describes how to combine SVMs built by bagging and the random subspace method for retrieving event shots.

When matching x with *rule*, we regard decision boundaries of SVMs as inaccurate, as described above. Decision rule matching is conducted based on probabilistic outputs of SVMs. To explain this, SVM_{l^*} is used to represent the l^* -th SVM corresponding to each feature a_{l^*} in *rule* ($1 \leq l^* \leq L$). In addition, the probabilistic output of the SVM_{l^*} for x is denoted by $Prob_{SVM_{l^*}}(x)$. Based on the above notations, we calculate $Match(x, rule)$ which is an evaluation value of matching x with *rule*.

$$Match(x, rule) = \prod_{l^*=1}^L Prob_{SVM_{l^*}}(x) \tag{5}$$

$Match(x, rule)$ is the joint probability of L SVMs in *rule*. It should be noted that a threshold is required to determine whether or not x matches *rule*. In view of this,

²This simplification is achieved by using the distributive law $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ and the absorption law $A \vee (A \wedge B) = A$. Although the simplification of a Boolean function is NP-hard, we can obtain an approximate solution by using the genetic algorithm [13].

the following two considerations must be made. Firstly, feature numbers (i.e. L) differ depending on decision rules. Secondly, the distribution of SVM's probabilistic outputs differs depending on the sigmoid function estimated for the SVM. For example, the mean of some SVM's probabilistic outputs is 0.5, while the one of a different SVM's probabilistic outputs is 0.3. Thus, it is unreasonable to use the same threshold for all decision rules.

Instead of actual values of $Match(x, rule)$, the following ranking approach is used to determine whether or not x matches $rule$. First, all shots are ranked in descending order of $Match(x, rule)$. A shot ranked within the top T -th position is considered to match $rule$ (T is set to 1,000 in all experiments). In this way, shots are matched with all decision rules based on the same criterion of the ranking position. Finally, our method outputs a retrieval result consisting of T' shots which match the largest numbers of decision rules ($T' = 1,000$ in our experiments).

3.2 Effectiveness of bagging and the random subspace method

In this section, we discuss whether bagging and the random subspace method are effective in extending the range of event shots that can be retrieved. That is, we examine differences among classification results of SVMs, which are built using different examples and feature dimensions.

Table 3 shows experimental results obtained for five events that will be used in Section 4. In particular, the objective is to examine whether classification results of SVMs change when using different examples and dimensions, even for the same type of feature. Thus, SVMs are built only by using *SIFT* feature. In the second to fourth rows of Table 3, we only use bagging where SVMs are built using the same number of randomly selected examples. On the other hand, in the fifth to seventh rows of Table 3, we use both bagging and the random subspace method where SVMs are built using the same number of randomly selected examples and *SIFT* dimensions. A comparison is drawn among classification results of 10 SVMs by examining 1,000 shots with the highest probabilistic outputs among the total 97,150 shots. We define the first classification result as the *baseline*, and examined the difference between the baseline and the remaining nine results, called *comparison results*. The second

Table 3 Differences among classification results of SVMs built using bagging and the random subspace method

Event		Event 1	Event 2	Event 3	Event 4	Event 5
Bagging	<i>Baseline</i> : # of event shots	124	99	111	35	146
	<i>Comparison</i> : # of different shots	466.9	579.7	470.1	396.6	512.8
	<i>Comparison</i> : # of different event shots	39.4	16.7	34.3	7.3	42.2
Bagging & random subspace	<i>Baseline</i> : # of event shots	129	97	115	22	150
	<i>Comparison</i> : # of different shots	461.3	695.4	656.4	428.4	693
	<i>Comparison</i> : # of different event shots	27.1	16.7	37.2	11.9	53.7

(or fifth) row represents the number of event shots included in the baseline. The third (or sixth) row represents the average number of shots that are only included in comparison results. Lastly, the fourth (or seventh) row represents the average number of event shots included only in comparison results.

As can be seen from Table 3, by changing examples with bagging, comparison results include approximately 397 to 580 shots that differ from the baseline. In addition, compared to event shots in the baseline, about 16 to 32% of different event shots are included in comparison results. Furthermore, by changing both examples and feature dimensions with bagging and the random subspace method, comparison results include about 428 to 695 shots that are different from the baseline, and approximately 17 to 36% of event shots in comparison results are different from event shots in the baseline. This indicates that bagging and the random subspace method are effective in extending the range of event shots that can be retrieved. However, Table 3 indicates that as the number of event shots which can be retrieved increases, the number of non-event shots also increases. It can be seen that the ratio of event shots to those shots included only in comparison results is less than 10%. Thus, in order to accurately retrieve event shots, decision rules need to be extracted as combinations of SVMs by using RST.

3.3 Partially supervised learning

Since the proposed RST relies on SVMs built using bagging and the random subspace method, it is necessary to collect n -examples that are useful for building accurate SVMs. In particular, considering the class imbalance problem, the proposed partially supervised learning (PSL) method should be able to collect a small number of *informative* n -examples from unlabeled examples (u -examples). N -examples similar to p -examples are considered as informative, because they are useful for characterizing the boundary between event and non-event shots. The procedure involved in selecting a small number of informative n -examples is described below.

The proposed PSL method is summarized in Algorithm 1. Firstly, since the number of event shots included in u -examples is very small, all u -examples are assumed to be n -examples. The set of p -examples and the set of n -examples are denoted by P and N , respectively. The number of p -examples and the one of n -examples are represented by $|P|$ and $|N|$, respectively. Based on P and N , an SVM, which examines whether or not n -examples are informative, is built. However, only a small number of n -examples can be used due to the class imbalance problem. If n -examples are randomly selected from N , n -examples located in certain regions of the feature space may not be selected. As a result, the decision boundary of the SVM is wrongly estimated, and it is not possible to appropriately evaluate the informativeness of n -examples. Thus, we need to collect a set of *representative* n -examples, which characterize the distribution of all n -examples. To this end, n -examples are grouped into clusters using the k -means clustering algorithm and the Euclidian distance measure. As shown in the line 1 of Algorithm 1, n -examples are grouped into N/β clusters. β is a pre-defined parameter used to control the number of clusters relative to the number of n -examples. Since various semantic contents are presented in n -examples, their features are very diverse. This necessitates many clusters of n -examples. In our experiment, β is set to 10, so that when $|N| = 30,000$, one can obtain 3,000 clusters. In addition, since it is difficult to appropriately measure

Algorithm 1 An overview of the proposed partially supervised learning

INPUT: set of p-examples P , set of n-examples N , desirable number of n-examples α , ratio between n-examples and clusters β
 OUTPUT: shrunk set of n-examples N

repeat

1. Cluster N into N/β clusters
2. Obtain the set of representative n-examples RN , each of which is centrally located in a cluster
3. Build an SVM using P and RN
4. From N , remove those n-examples distant from the decision boundary of the SVM

until $|N| \leq \alpha$ OR no n-example is removed from N
 5. Output N

similarities among n-examples in the 6,000-dimensional feature space defined in Fig. 4, our PSL method is conducted on 1,000-dimensional *SIFT* feature (the SVM in the line 3 of Algorithm 1 is also built on *SIFT* feature).

After clustering, for each cluster c , the most centrally located n-example is selected as the representative n-example n_c .

$$n_c = \min_{n_i \in N_c} \sum_{n_j \in N_c} dist(n_i, n_j) \tag{6}$$

where N_c is the set of n-examples in c , and n_i and n_j are n-examples in N_c . $dist(n_i, n_j)$ represents the Euclidian distance between n_i and n_j . Thus, n_c is selected as the n-example having the minimum sum of Euclidian distances to the other n-examples in c . A set of representative n-examples for all clusters is denoted by RN .

By using an SVM built on P and RN , it can be determined whether each n-example n in N is informative based on its distance from the decision boundary of the SVM. N-examples distant from the decision boundary are uninformative for defining the boundary between event and non-event shots. The above test is conducted using the following criterion:

$$|w^T n + b| > \gamma \tag{7}$$

Using x as an arbitrary example, $w^T x + b = 0$ represents the decision boundary (hyperplane) of the SVM. Based on this, $|w^T n + b|$ can characterize the distance between n and the decision boundary. Specifically, the distance is $|w^T n + b|/||w||$, but since $||w||$ is constant, it can be omitted. If the distance between n and the decision boundary is larger than the threshold γ , n is regarded as uninformative and subsequently removed from N . This removal of uninformative n-examples is iterated until the number of n-examples is less than the pre-defined number of n-examples α or no further n-examples are removed from N .

An example illustrating the above iteration is shown in Fig. 7, with circles and crosses representing p-examples and n-examples, respectively. N-examples are initially collected as shown in Fig. 7a, and the first iteration is then performed as shown in Fig. 7b. N-examples are grouped into four clusters and an SVM is built using

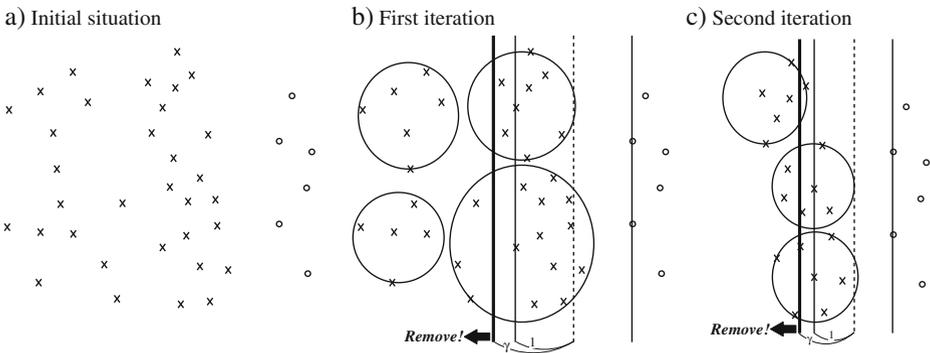


Fig. 7 An example of the proposed partially supervised learning method

representative n -examples from these clusters. As a result, two p -examples and two representative n -examples are extracted as support vectors, as depicted by the solid lines. The dashed line represents the decision boundary of the SVM. Based on this, n -examples located on the left side of the bold line are regarded as uninformative and are subsequently discarded. The second iteration is then performed using the remaining n -examples, as shown in Fig. 7c. In this iteration, an SVM is built using three representative n -examples. Based on the decision boundary of this SVM, n -examples located on the left side of the bold line are discarded. In this way, a small number of n -examples, which are highly similar to p -examples, can be obtained. Such n -examples are useful for characterizing the boundary between event and non-event shots.

4 Experimental results

In this section, the proposed QBE method is tested on TRECVID 2009 video data [29]. This data consists of 219 development and 619 test videos in various genres, like cultural, news magazine, documentary and education programming. Each video is already divided into shots by using an automatic shot boundary detection method, where development and test videos include 36,106 and 97,150 shots, respectively (the detailed video data specification can be found on TRECVID 2009 Web site³). The proposed method is evaluated based on the following five events:

- Event 1:* A view of one or more tall buildings and the top story visible
- Event 2:* Something burning with flames visible
- Event 3:* One or more people, each at a table or desk with a computer visible
- Event 4:* An airplane or helicopter on the ground, seen from outside
- Event 5:* One or more people, each sitting in a chair, talking

Each event retrieval is conducted as follows. Firstly, p -examples are manually collected from development videos. Based on p -examples, n -examples are then collected using the proposed PSL method. Considering the class imbalance problem,

³<http://www-nlpir.nist.gov/projects/tv2009/tv2009.html>

the number of collected n-examples is set to be equal to five times the number of p-examples. Afterward, the proposed RST method extended by bagging and the random subspace method, is run where two libraries, LIBSVM [9] and ROSETTA [13], are used for SVM learning and for the reduct extraction in RST, respectively. SVM parameters are determined using 3-fold cross validation. Lastly, the retrieval performance is evaluated based on the number of event shots within 1,000 retrieved shots.

4.1 Effectiveness of rough set theory extended by bagging and the random subspace method

To examine the effectiveness of RST extended by bagging and the random subspace method, a comparison is drawn among the four types of retrieval described below.

- Baseline:* For each of six features, an SVM is built using all examples and dimensions, and a search of test videos is conducted. The SVM which yields the best result is then manually selected. In other words, *Baseline* represents a favorable retrieval result under the ideal condition in which the best feature for the event can be selected.
- RST_only:* RST, which uses neither bagging nor the random subspace method, is executed. One SVM is built for each feature using all examples and dimensions.
- RST+BG:* RST is executed only by using bagging. For each feature, three SVMs are built using different subsets of examples and all dimensions. Each subset is constructed by randomly sampling 75% of examples.
- RST+BG+RS:* RST which incorporates both bagging and the random subspace method is executed. For each feature, 10 SVMs are built using different subsets of examples and dimensions. Each subset of examples is formed by randomly sampling 75% of examples, while each subset of dimensions is formed by randomly sampling 50% of dimensions.

Table 4 shows performances of the above four types of retrieval. For each event, the second row presents the number of p-examples. For the performance evaluation, we consider that retrieval results of the proposed method differ due to the following two random factors. The first is attributed to the fact that, the PSL method often terminates before the number of n-examples is reduced to the specified number, because there are no n-examples which can be filtered out (refer to the stopping criterion in Algorithm 1). In such a case, from the remaining n-examples, five times as many n-examples as p-examples are randomly selected. The second random factor is associated with bagging and the random subspace method, where examples and feature dimensions are randomly selected. Thus, in Table 4, each row labeled ‘# of event shots’ indicates the mean number of event shots in 10 retrieval results. Similarly, rows labeled ‘# of decision rules’ and ‘# of average precision’ indicate the mean number of decision rules extracted by RST and the mean of average precisions, respectively.

Table 4 Performance comparison of *Baseline*, *RST_only*, *RST+BG* and *RST+BG+RS*

	Event	Event 1	Event 2	Event 3	Event 4	Event 5
	# of p-examples	100	46	61	40	124
<i>Baseline</i>	# of event shots	161.1	98.9	163.0	42.7	151.0
<i>RST_only</i>	# of event shots	165.1	111.0	161.5	34.1	146.9
	# of decision rules	5.5	5.2	5.4	4.5	3.8
	Average precision	0.0931	0.1449	0.0871	0.0094	0.0365
<i>RST+BG</i>	# of event shots	170.1	137.3	176.7	38.2	196.4
	# of decision rules	252.9	159.7	151.9	137.4	354.0
	Average precision	0.1148	0.1827	0.0877	0.0154	0.0571
<i>RST+BG+RS</i>	# of event shots	172.0	147.5	176.9	41.6	194.6
	# of decision rules	6159.2	1797.6	2286	1822.9	11455.6
	Average precision	0.1170	0.1942	0.0992	0.0159	0.0604

In Table 4, the numbers in bold fonts indicate that *RST_only*, *RST+BG* or *RST+BG+RS* can retrieve more event shots than *Baseline*. Since the performance of *RST_only* for *Event 3*, *Event 4* and *Event 5* is lower than that of *Baseline*, it is not necessarily effective. One main reason for this ineffectiveness of *RST_only* is that the number of extracted decision rules for each event is very small. On the other hand, except for *Event 4* (which will be discussed later), both *RST+BG* and *RST+BG+RS* outperform *Baseline* where a greater number of decision rules are extracted compared to *RST_only*. Thus, bagging and the random subspace method are useful for building various SVMs, which enables the extraction of decision rules covering a large variety of event shots. Lastly, when making a comparison between *RST+BG* and *RST+BG+RS*, numbers of retrieved event shots are not significantly different from each other. For all events, average precisions of *RST+BG+RS* are higher than those of *RST+BG*, implying that event shots are ranked at higher positions in a retrieval result by *RST+BG+RS* than by *RST+BG*. Thus, *RST+BG+RS* is considered to be superior to *RST+BG*.

For *Event 4*, one main reason for the ineffectiveness of *RST+BG* and *RST+BG+RS* is the difficulty of accurately recognizing airplanes and helicopters. Specifically, SVMs built by bagging and the random subspace method wrongly classify many shots showing cars, trains, ships etc. as positive, because their shapes are relatively similar to those of airplanes and helicopters. Thus, combining such inaccurate SVMs into decision rules degrades the retrieval performance.

4.2 Effectiveness of partially supervised learning

In this section, we examine the effectiveness of our PSL method. To this end, the performance using n -examples collected by our PSL method is compared to the one using n -examples which are randomly collected from all of u -examples. For the simplicity, we call the former and the latter types of n -examples *PSL* n -examples and *random* n -examples, respectively. In Table 5, we run *RST+BG* and *RST+BG+RS* using the same p -examples in Table 4. PSL n -examples are used in the second and third rows while random n -examples are used in the fourth and fifth rows. Each

Table 5 Comparison between the retrieval performance using our PSL method and the one using the random n-example selection

Event		Event 1	Event 2	Event 3	Event 4	Event 5
<i>PSL</i>	<i>RST+BG</i>	170.1	137.3	176.7	38.2	196.4
	<i>RST+BG+RS</i>	165.1	147.5	176.9	41.6	194.6
<i>Random</i>	<i>RST+BG</i>	157.9	136.0	168.5	46.6	187.5
	<i>RST+BG+RS</i>	159.6	144.3	171.0	47.0	187.8

performance in Table 5 is evaluated as the average number of event shots in 10 retrieval results.

As can be seen from Table 5, except for *Event 4*, event retrieval using PSL n-examples is more accurate than the one using random n-examples. The reason for the ineffective performance in *Event 4* is as follows. Due to the difficulty of accurately recognizing helicopters and airplanes, few edges which characterize the sky and runways in the background, are important for *Event 4*. But, since PSL n-examples are highly similar to p-examples, most of them are characterized by few edges. So, several event shots with few edges are inevitably excluded from the retrieval result. Thus, it can be said that our PSL method works well for events involving objects, which can be relatively accurately recognized. For example, for *Event 1*, the net of a soccer goal, iron bars, closet etc. are shown in PSL n-examples, because shapes of these objects are similar to buildings. By comparing such PSL n-examples to p-examples, a precise boundary between event and non-event shots can be extracted. Finally, since random n-examples are currently used in almost all of state-of-the-art methods [21, 22, 31], we believe that our PSL method is novel in the sense that it can outperform the random n-example selection.

4.3 Effectiveness for the small sample size problem

The performance of the proposed method is also tested in the case where only a small number of p-examples are available. Figure 8 illustrates the difference in the retrieval performance of *Baseline*, *RST+BG* and *RST+BG+RS*, depending on p-example numbers. For a specified p-example number, we construct three different sets of examples in the following way. Firstly, the specified number of p-examples are randomly collected from all available p-examples, as shown in Table 4. Then, n-examples are collected using our PSL method. The performance is evaluated as the average number of event shots in retrieval results using the above three sets of examples.

In Fig. 8, for *Event 2* and *Event 3*, *RST+BG* and *RST+BG+RS* always outperform *Baseline*. For *Event 1* and *Event 5*, when only 10 p-examples are available, *RST+BG* and *RST+BG+RS* are outperformed by *Baseline*. In this case, most of SVMs are inaccurate and as a consequence, decision rules as combinations of these SVMs are also inaccurate. Finally, for *Event 4*, *RST+BG* and *RST+BG+RS* are always outperformed by *Baseline*, due to the difficulty of accurately recognizing airplanes and helicopters described in the previous section. To summarize the overall performance, except for *Event 4*, *RST+BG* and *RST+BG+RS* become more appropriate than *Baseline* when greater than 20 p-examples are available.

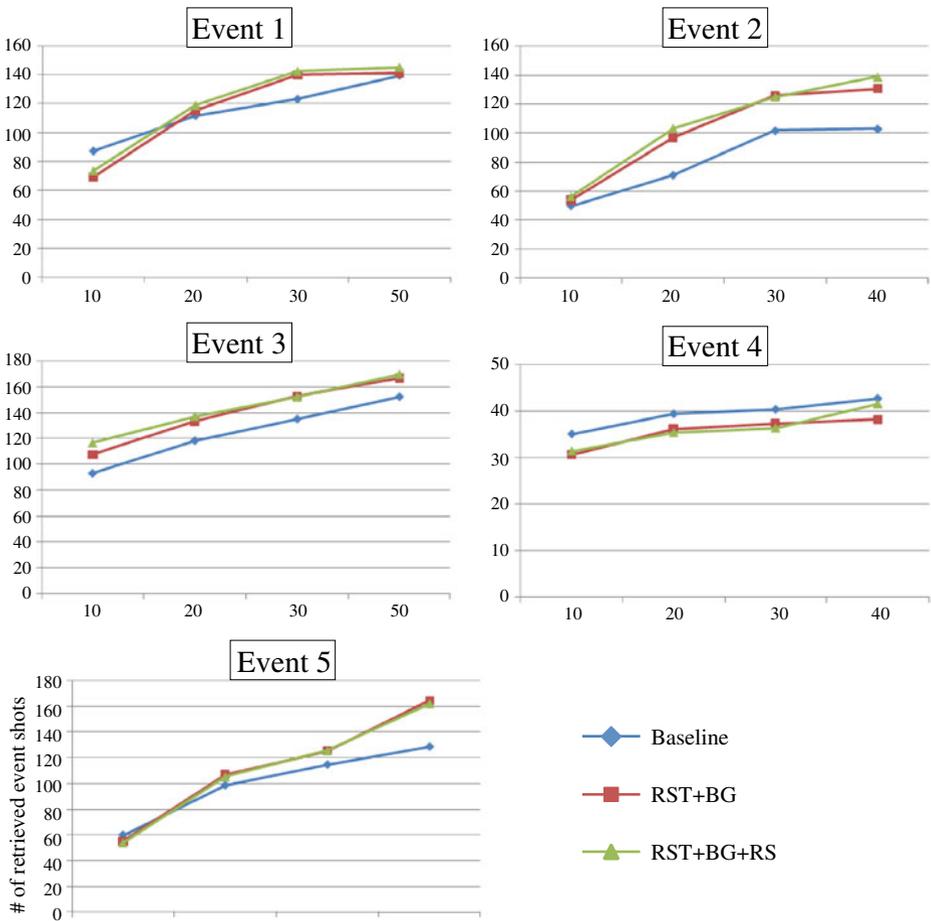


Fig. 8 Retrieval performances for different available p-example numbers

Finally, it is easy to collect more than 20 p-examples for *Event 1* and *Event 5* as event shots of these are often seen in videos. However, event shots of *Event 2* are rarely seen and there are only a limited number of videos containing these event shots. This increases the difficulty of collecting more than 20 p-examples for *Event 2*. A sufficient number of p-examples may be obtained by retrieving images and videos on the web using online image/video search engines such as Flickr and YouTube.

4.4 Performance comparison

A comparison is made between the performance of the proposed method and those of state-of-the-art methods. *RST+BG+RS* is specially compared with methods developed in TRECVID 2009 search task [29]. This task consists of three categories, namely, the fully automatic, manually-assisted and interactive categories. Given the textual description of an event and some p-examples, methods in the fully automatic category retrieve event shots without any user intervention. In the manually-assisted

category, a user intervention is allowed only prior to the start of the test video search. The interactive category allows an interactive user intervention based on retrieval results. Although our proposed method belongs to the manually-assisted category, only three retrieval results are submitted to this category. This is clearly insufficient for achieving a meaningful comparison. Thus, the retrieval result of the proposed method is compared with 88 results in the fully automatic category.

Figure 9 shows the maximum (solid arrow) and average (dashed arrow) numbers of event shots among 10 retrieval results, obtained using *RST+BG+RS* in Table 4. As can be seen from Fig. 9, the overall performance of the proposed method is ranked in the top quartile. It can be noted that almost all methods in the top quartile use classifiers to assess the relevance of each shot to keywords like *Person*, *Building*, *Cityspace* and so on. In this case, for each keyword, a classifier is built using a large number of training examples. For example, in the method proposed by researchers at City University of Hong Kong, classifiers for 374 keywords are built using 61,901 manually annotated shots [22]. Also, in the method developed by researchers at

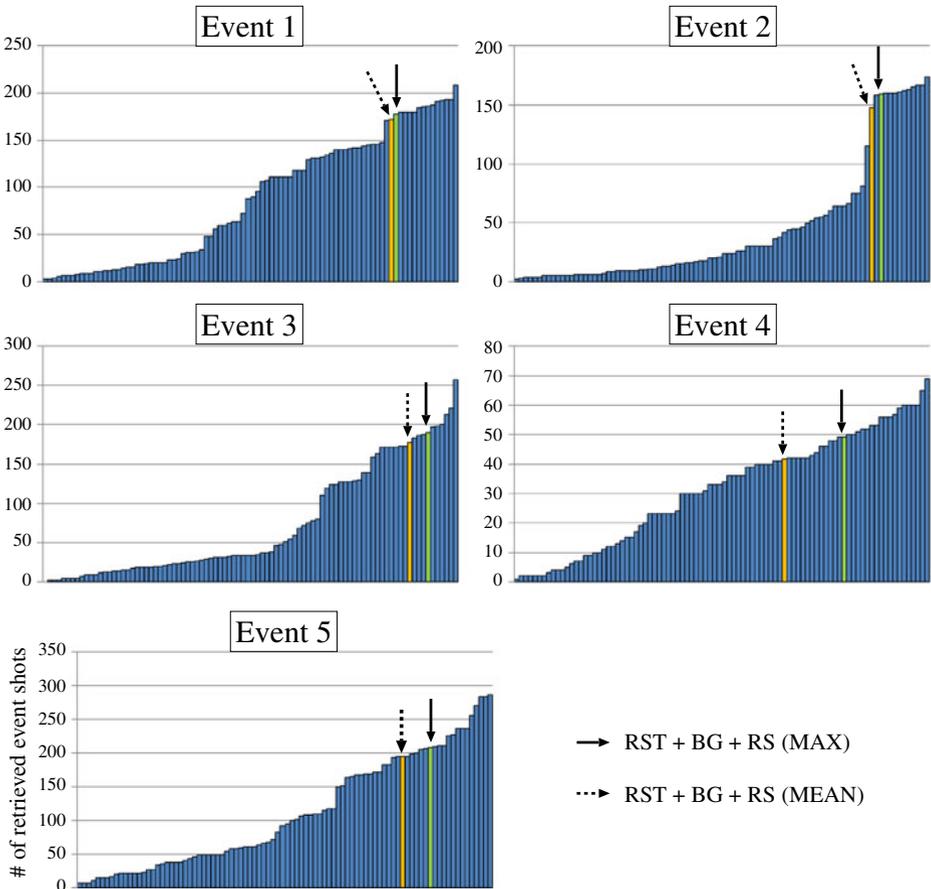


Fig. 9 Performance comparison between our method and methods in TRECVID 2009

University of Amsterdam, classifiers for 64 keywords are built using more than 10,000 manually annotated shots, such as 39,674 shots for *Bus*, 21,532 shots for *Car* and so on [31]. Like this, methods in the top quartile require tremendous manual effort. Compared to these methods, our method only uses p-examples which are provided by a user in an off-the-cuff manner. Thus, it can be concluded that the proposed method is very effective in the sense that it requires neither manual shot annotation nor classifier preparation.

4.5 Reducing computation cost by parallelization

In this section, we examine the computation cost of our QBE method and reduce it by parallelizing several processes. Our method consists of two main phases, PSL and RST. Figure 10 illustrates processes in PSL (a) and RST (b) phases. Roughly speaking, the input of the PSL phase is a set of p-examples, and its output is a set of n-examples which are as similar to p-examples as possible. To this end, we first regard all of u-examples as n-examples and group them into clusters. Secondly, we find the representative n-example for each cluster. An SVM is then built by using p-examples and representative n-examples. Subsequently, n-examples which are distant from the decision boundary of the SVM are removed. Finally, the above processes are iterated until the number of n-examples is less than the pre-defined threshold or no n-example can be removed.

When a large number of n-examples remain, the following two processes require expensive computation costs. The first is the similarity calculation for calculating the similarity between an n-example and each cluster center. The second computationally expensive process is cross validation where, for each parameter candidate, an SVM is built and evaluated by computing the error rate in classifying p-examples and n-examples. Thus, we parallelize the above two processes by using a multicore

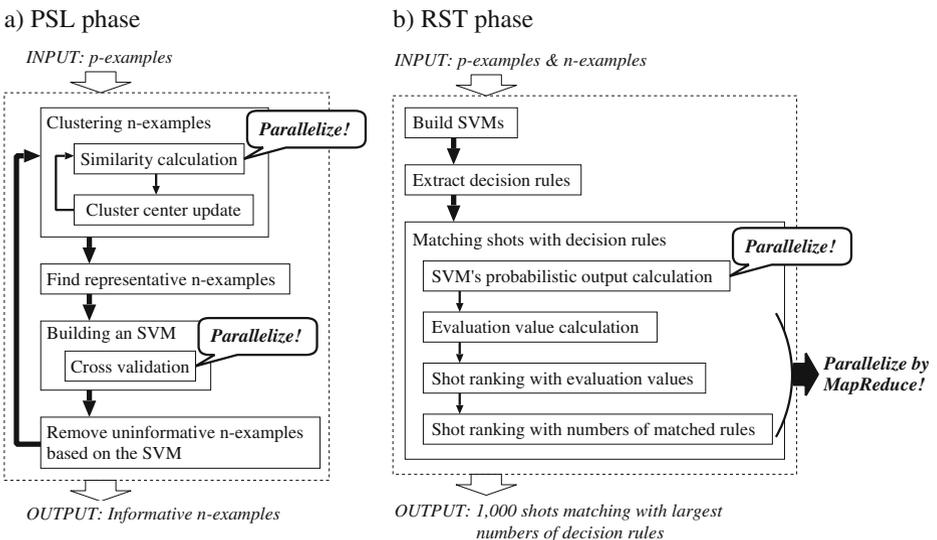


Fig. 10 Illustrations of processes in PSL (a) and RST (b) phases

PC. For the similarity calculation process, each core is used to calculate the similarity between one cluster center and n -examples. For the cross validation process, each core is used to compute the error rate of an SVM with one parameter candidate. These are implemented using Matlab Parallel Computing Toolbox [19].

As shown in Fig. 10b, the input of the RST phase is a set of p -examples and n -examples, and its output is the set of 1,000 shots which match the largest numbers of decision rules. The RST phase is summarized as follows. Firstly, various SVMs are built by using bagging and the random subspace method. Decision rules are then extracted based on SVMs' classification results of p -examples and n -examples. Subsequently, in order to match shots with extracted decision rules, the following four processes are performed. The first one is SVM's probabilistic output calculation to compute the probabilistic output of an SVM for each shot. The second process is the evaluation value calculation which computes the evaluation value of matching a shot with each decision rule. The third is shot ranking with evaluation values where, for each decision rule, shots are ranked in the descending order of evaluation values. Based on this, 1,000 shots with the largest evaluation values are regarded to match the decision rule. The final process is shot ranking with matched rules to determine 1,000 shots matching the largest numbers of decision rules.

The RST phase has a high computation cost due to a large number of shots and decision rules. We address this by using two types of parallelizations on a multicore PC. The first one is applied to the process of SVM's probabilistic output calculation, where each core is used to compute SVM's probabilistic output for a distributed set of shots. The other three processes are implemented by using *MapReduce*, which is a parallel programming model that provides a simple and powerful interface [24]. We use 'Phoenix' which is a MapReduce library for multicore PCs, in order to save a significant amount of time on I/O operations [24]. In MapReduce, the basic data structure is a *(key, value)* pair. Based on this, the *Map* function constructs input *(key, value)* pairs from a distributed data, and produces intermediate *(key, value)* pairs by conducting a user-defined task. Subsequently, the *Reduce* function conducts a user-defined merge operation on intermediate *(key, value)* pairs with the same *key* and outputs a final result. In this manner, MapReduce divides a large-scale data into small pieces of *(key, value)* pairs, which are efficiently processed in parallel by the Map and Reduce functions.

Three processes in Fig. 10b are implemented by utilizing MapReduce twice. The first MapReduce performs two processes, the evaluation value calculation and shot ranking with evaluation values. Specifically, the objective is to determine 1,000 shots which have the largest evaluation values for each rule. To do this, the following Map and Reduce functions are designed:

$$\begin{aligned} \text{map}_1 &: (x, [\text{rule}, \text{Prob}_{SVM}^{\text{all}}(x)]) \rightarrow \text{List}(\text{rule}, \text{Match}(x, \text{rule})) \\ \text{reduce}_1 &: (\text{rule}, \text{List}(\text{Match}(x, \text{rule}))) \rightarrow (\text{rule}, \text{SList}_{\text{rule}}^{1,000}) \end{aligned} \quad (8)$$

where x and *rule* are a shot and decision rule, respectively. $\text{Prob}_{SVM}^{\text{all}}(x)$ represents the set of probabilistic outputs of all SVMs for x . By using $\text{Prob}_{SVM}^{\text{all}}(x)$, map_1 computes $\text{Match}(x, \text{rule})$ which is the evaluation value of matching x with *rule* as defined in (5). The output of map_1 is a list including evaluation values of distributed shots for all rules. Subsequently, reduce_1 merges such lists produced by map_1 on different cores so that for the same *rule*, evaluation values of all shots are combined into a list,

namely, $List(Match(x, rule))$. By sorting this list, $reduce_1$ outputs $SList_{rule}^{1,000}$ which consists of 1,000 shots with the largest evaluation values. That is, shots in $SList_{rule}^{1,000}$ are regarded to match $rule$.

The second MapReduce performs the process of shot ranking with numbers of matched rules. This process aims to obtain 1,000 shots which match the largest numbers of decision rules. The following Map and Reduce functions are designed:

$$\begin{aligned}
 map_2 &: (x, 1) \rightarrow List(x, MRules(x)) \\
 reduce_2 &: (x, List(MRules(x))) \rightarrow SList_{MRules}^{1,000}
 \end{aligned}
 \tag{9}$$

where $(x, 1)$ is obtained by parsing $SList_{rule}^{1,000}$ and indicates that a shot x matches one rule. It should be noted that as we have only to count the number of decision rules matched with x , we do not need to know which rules are matched with x . The function map_2 constructs a list of $(x, MRules(x))$ s, where $MRules(x)$ represents the number of decision rules matched with x . Subsequently, $reduce_2$ merges $MRules(x)$ for the same shot into $List(MRules(x))$, which represents the total number of matched decision rules. Finally, the function sorts all shots based on $List(MRules(x))$ and outputs $SList_{MRules}^{1,000}$ which consists of 1,000 shots matching the largest numbers of decision rules.

Figure 11 illustrates retrieval times for our QBE method where event shots for *Event 1* are retrieved by applying *RST+BG* to 100 p-examples and 500 n-examples. Figure 11a and b show the change in computation times in the PSL and RST phases, respectively. In both figures, the four bars from the top to the bottom represent computation times by parallelizing our method with 1, 2, 4 and 8 cores, respectively. Bars named *Others* depict computation times of non-parallelized processes while the other bars depict computation times of parallelized processes. Specifically, in the PSL phase in Fig. 11a, *Clustering* and *SVM building* include the similarity calculation and cross validation processes in Fig. 10a, respectively. In the RST phase in Fig. 11b, *SVM prob.* corresponds to SVM’s probabilistic output calculation process in Fig. 10b. *Rule matching* includes three processes parallelized by MapReduce. In addition, numbers overlaid with bars present actual values of computation times. Both of Fig. 11a and b indicate that, as the number of cores increases, the computation time can be significantly shortened. Specifically, when only one core is used, our QBE method takes a total of 26,312 seconds to complete event retrieval with the PSL and RST phases requiring 24,796 and 1,516 seconds, respectively. In comparison, when 8 cores

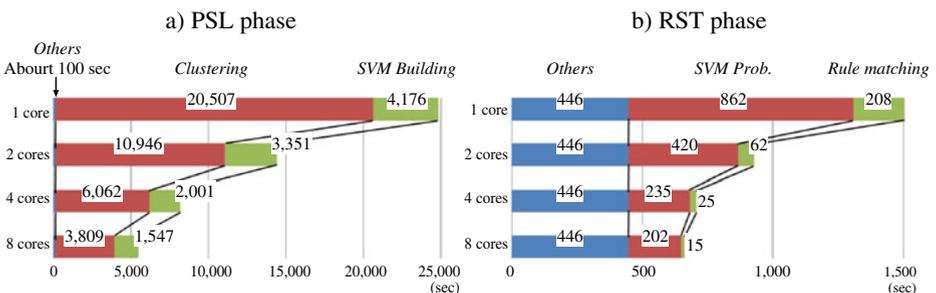


Fig. 11 Comparison among retrieval times by parallelizing our QBE method with 1, 2, 4 and 8 cores

are used, event retrieval is completed in 6,194 seconds with computation times of the PSL and RST phases being 5,531 and 663 seconds, respectively.

From the perspective of computation time, our current QBE method is far from the satisfactory. So, the retrieval time will need to be further reduced by improving currently parallelized processes and parallelizing the other processes. In relation to this concern, Fig. 11 reveals an important issue. For *Clustering* and *Rule matching* processes, as the number of cores increases, computation times are nearly linearly reduced. However, compared to these reductions, reductions of computation times for *SVM building* and *SVM prob.* are significantly less effective. For example, for *SVM building* process in the PSL phase, even if the number of cores is doubled from 4 to 8, the ratio between the computation time of 4 cores and the one of 8 cores is only 1.29 (i.e. 2,001 seconds versus 1,547 seconds). One consideration is that *SVM building* and *SVM prob.* are parallelized simply by distributing examples (or shots) to multiple cores. On each core, functions in LibSVM libraries [9] are called where memory allocations and releases are executed many times. This degrades the effectiveness of parallelization. Therefore, in order to achieve effective parallelization of a process, it should be divided into sub-processes involving few memory allocations or releases.

5 Conclusion and future work

In this paper, we proposed a QBE method which can retrieve event shots using only a small number of p-examples. Due to camera techniques and settings, event shots are characterized by significantly different features. As such, RST is used to extract multiple decision rules which characterize different subsets of p-examples. A variety of event shots can be retrieved where each decision rule is specialized to retrieve a part of event shots. Additionally, in order to extend the range of event shots that can be retrieved, bagging and the random subspace method are incorporated into RST. Classifiers built using different examples and feature dimensions, are useful for covering a variety of event shots while many non-event shots are potentially retrieved. Thus, RST is used to combine classifiers into decision rules in order to accurately retrieve event shots. Furthermore, to overcome the lack of n-examples, PSL is used to collect n-examples from u-examples. In particular, taking the class imbalance problem into account, a method which can collect a small number of n-examples useful for building an accurate classifier, is developed. Experimental results demonstrated that our method successfully covers various event shots when more than 20 p-examples are available. In addition, our method can achieve very effective event retrieval without any shot annotation or classifier preparation.

The following issues will be addressed in future works. Firstly, we aim to use temporal features such as 3DSIFT [27] and acoustic features such as Mel-Frequency Cepstrum Coefficient (MFCC), as opposed to our current method which only makes use of image features. Secondly, although majority voting is currently conducted using all of extracted decision rules, some rules may be inaccurate or very similar to other rules. Thus, in order to obtain the optimal set of decision rules, a method which examines the accuracy of each decision rule based on cross validation will be developed. The relationship among decision rules will be investigated using the diversity measures proposed in [14]. Lastly, to improve the computation time of our current method, we plan to build a cluster consisting of tens or hundreds of PCs. On this cluster, in addition to the process of matching shots with decision rules,

we parallelize the other processes using Apache Hadoop [35], which implements MapReduce on a large-scale PC cluster.

Acknowledgements 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, Kwek S, Japkowicz N (2004) Applying support vector machines to imbalanced datasets. In: Proc. of ECML 2004, pp 39–50
2. Alpaydin E (2004) Introduction to machine learning. MIT Press
3. Breiman L (1996) Bagging predictors. *Mach Learn* 24(2):123–140
4. Fung G, Yu J, Ku H, Yu P (2006) Text classification without negative examples revisited. *IEEE Trans Knowl Data Eng* 18(1):6–20
5. Gang Z, Kobayashi A, Sakai Y (2005) Group-based relevance feedback using reduct of rough set theory for interactive image retrieval. *J ITEJ* 56(6):884–893
6. Guo G, Dyer C (2005) Learning from examples in the small sample case: face expression recognition. *IEEE Trans Syst Man Cybern Part B* 35(3):477–488
7. Han J, Kamber M (2006) Data mining: concepts and techniques, 2nd edn. Morgan Kaufmann Publishers
8. Ho T (1998) The random subspace method for constructing decision forests. *IEEE Trans Pattern Anal Mach Intell* 20(8):832–844
9. Hsu C, Chang C, Lin C (2003) A practical guide to support vector classification. <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
10. Jain A, Duin R, Mao J (2000) Statistical pattern recognition: a review. *IEEE Trans Pattern Anal Mach Intell* 22(1):4–37
11. Japkowicz N (2000) The class imbalance problem: significance and strategies. In: Proc. of IC-AI 2000, pp 111–117
12. Kashino K, Kurozumi T, Murase H (2003) A quick search method for audio and video signals based on histogram pruning. *IEEE Trans Multimedia* 5(3):348–357
13. 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
14. Kuncheva L, Whitaker C (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach Learn* 51(2):181–207
15. Lin W, Hauptmann A (2004) Modeling timing features in broadcast news video classification. In: Proc. of ICME 2004, pp 27–30
16. Liu B, Lee W, Yu P, Li X (2002) Partially supervised classification of text documents. In: Proc. of ICML 2002, pp 387–394
17. Liu H (2005) Evolving feature selection. *IEEE Intell Syst* 20(6):64–76
18. Liu H, Lin C, Weng R (2007) A note on Platt's probabilistic outputs for support vector machines. *Mach Learn* 68(3):267–276
19. Luszczek P (2008) Enhancing multiple system performance using parallel computing with MATLAB. <http://www.mathworks.com/mason/tag/proxy.html?dataid=10721&fileid=55087>
20. Mizui A, Shirahama K, Uehara K (2008) TRECVID 2008 NOTEBOOK PAPER: interactive search using multiple queries and rough set theory. In: Proc. of TRECVID 2008, pp 123–132
21. 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
22. 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
23. Peng Y, Ngo C (2005) EMD-based video clip retrieval by many-to-many matching. In: Proc. of CIVR 2005, pp 71–81

24. Ranger C, et al (2007) Evaluating mapreduce for multi-core and multiprocessor systems. In: Proc. of HPCA 2007, pp 13–24
25. Saha S, Murthy C, Pal S (2007) Rough set based ensemble classifier for web page classification. *Fundam Math* 76(1–2):171–187
26. 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
27. 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
28. Shirahama K, Sugihara C, Matsuoka Y, Matsumura K, Uehara K (2009) Kobe University at TRECVID 2009 search task. In: Proc. of TRECVID 2009, pp 76–84
29. Smeaton A, Over P, Kraaij W (2006) Evaluation campaigns and TRECVID. In: Proc. of MIR 2006, pp 321–330
30. Snoek C, Worring M (2005) Multimedia event-based video indexing using time intervals. *IEEE Trans Multimedia* 7(4):638–647
31. Snoek C, et al (2009) The mediamill TRECVID 2009 semantic video search engine. In: Proc. of TRECVID2009, pp 226–238
32. Tao D, Tang X, Li X, Wu X (2006) Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Trans Pattern Anal Mach Intell* 28(7):1088–1099
33. Vapnik V (1998) *Statistical learning theory*. Wiley-Interscience
34. Verbeek J, Vlassis N, Kröse B (2003) Efficient greedy learning of gaussian mixture models. *Neural Comput* 15(2):469–485
35. White T (2009) *Hadoop: the definitive guide*. O'Reilly Media
36. Yao P (2009) Hybrid classifier using neighborhood rough set and svm for credit scoring. In: Proc. of BIFE 2009, pp 138–142
37. Yu H, Han J, Chang K (2004) PEBL: web page classification without negative examples. *IEEE Trans Knowl Data Eng* 16(1):70–81
38. 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



Kimiaki Shirahama received his B.E., M.E. and Ph.D. 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, 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 Matsuoka received his B.E. degree from Kobe University, Japan in 2009. He is currently pursuing his M.E. degree in the Graduate School of Engineering at Kobe University. His research interests include image/video data retrieval and machine learning. He is a member of the Institute of Electronics, Information and Communication Engineering in Japan (IEICE).



Kuniaki Uehara received his B.E., M.E. and Ph.D. 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 AAI, 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).