Relevance Feature Mapping for Content-Based Image Retrieval

Guang-Tong Zhou[1], Kai Ming Ting[2], Fei Tony Liu[2], Yilong Yin[1]
[1]School of Computer Science and Technology, Shandong University, Jinan 250101, China
[2]Gippsland School of Information Technology, Monash University, Victoria 3842, Australia
zhouguangtong@gmail.com, {kaiming.ting,tony.liu}@infotech.monash.edu.au, ylyin@sdu.edu.cn

ABSTRACT

This paper presents a ranking framework for content-based image retrieval using relevance feature mapping. Each relevance feature measures the relevance of an image to some profile underlying the image database. The framework is a two-stage process. In the off-line modeling stage, it constructs a collection of models which maps all images in the database to the relevance feature space. In the on-line retrieval stage, it assigns a weight to every relevance feature based on the query image, and then ranks images in the database according to their weighted average feature values. The framework also incorporates relevance feedback which modifies the ranking based on the feedbacks through reweighted features. We show that the power of the proposed framework is coming from the relevance features. Experiments on a large image database validate the efficacy and efficiency of the proposed framework.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval — Retrieval models

General Terms

Algorithms

1. INTRODUCTION

With the rapid increase of digital image collections, effective and efficient retrieval techniques become more and more important. Many existing image retrieval systems index and search the images based on textual information such as keywords, surrounding text, etc.. However, the text-based search suffers from the following inherent drawbacks [17, 4]: (i) the textual information is usually nonexistent or incomplete with the emergence of massive image databases; (ii) the textual description is not sufficient for depicting subjective semantics since different people may describe the content of an image in different ways; and (iii) some image contents are difficult to be described in words.

To address these problems, content-based image retrieval (CBIR) is proposed and has attracted a lot of research interest in recent years [15, 4]. In a typical CBIR setting, a user poses a query image to the system in order to retrieve relevant images from the database. However, due to the semantic gap [15] between high-level concepts and low-level features, the list returned by the initial search may not be good enough to satisfy the user's requirement. Thus, relevance feedback [14, 18] is usually employed to allow the user to iteratively refine the query information by labeling a few positive images as well as negative images in each feedback round.

The performance of a CBIR system mainly relies on the accuracy of its ranking results. Thus, ranking is the central problem in CBIR; and many have endeavored to design a fast and effective ranking method. A key ingredient in ranking is the measure used for comparing images in the database with respect to the query. Many existing methods use distance as the core ranking measure.

This paper presents a novel ranking framework for CBIR by using an ensemble of ranking models. The framework utilizes no distance or similarity measure, which is fundamentally different from most existing methods. Our framework makes use of ranking models to form a relevance feature space. It builds a collection of ranking models and the output of each model forms a relevance feature. Then, the models are used to map every database image into a point in a new space of relevance features. Finally, the ranking and retrieval of images, based on one query and relevance feedbacks, are computed in the new space. Our analysis shows that the power of the proposed framework comes from the relevance features. Our framework has linear time and space complexities w.r.t. the database size, and the on-line processing time is constant no matter how many original features are used to represent an image. These characteristics enable the proposed framework to scale up to high-dimensional CBIR tasks. Besides, our framework has a good tolerance for random features.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the proposed framework, followed by a detailed description in Section 4. Then, Section 5 reports the empirical studies, and Section 6 provides a discussion on related issues. Finally, this paper concludes in Section 7.

MDMKDD'10, July 25th, 2010, Washington, DC, USA.

2. RELATED WORK

Many ranking methods employ distance as the core ranking measure. In the case of retrieval with one query without relevance feedback, the majority of previous works have focused on defining or refining the distance. The simplest way is to use a single distance metric, e.g., Euclidean distance or Manhattan distance. Here images that lie near to a given query are ranked higher than images far away from the query. However, the above-mentioned distance metrics are global measures and they might not produce the best results for all queries, even though they are widely used. Thus, researchers have tried to find distance metrics that can be tailored to each query. For example, based on the manifold ranking algorithm, He et al. [7] proposed the `MRBIR` method which implicitly learns a manifold metric to produce ranking for each query.

In relevance feedback, the additional information provided by the user offers more flexibility in the design of effective ranking methods. Here the query and positive feedbacks are usually considered as positive images, and negative feedbacks are negative images. The refinement can be done in three ways. First, the distance metric for the initial query session can be further refined based on pair-wised distance constraints derived from positive and negative images. Commonly used techniques include distance metric learning [5], kernel learning [16], and manifold learning [8].

Second, apart from refining the distance metric, we can also tackle the problem by designing appropriate ranking schemes. For example, MARS (Multimedia Analysis and Retrieval System) [13] employs a query-point movement technique which estimates the "ideal query point" by moving it towards positive images and away from negative ones. Then the ranking is produced by measuring distance with respect to the ideal query. Giacinto and Roli [6] proposed the `InstRank` method based on the idea that an image is more likely to be relevant if its distance to the nearest positive image is small, while an image is more likely to be irrelevant if its distance from the nearest negative image is small. A recent method called `Qsim` [20] advocates ranking images based on the query-sensitive similarity measure, which takes into account the queried concept when measuring similarities. Note that this kind of methods is based on a predefined or learned distance metric.

Third, some methods transform the CBIR problem into a classification problem, and solve it using a classification technique such as support vector machine [11], and Bayesian method [17]. A representative method called `BALAS` [17] first estimates the probability density function of positive class as well as negative ones, and then produces the ranking using a Bayesian learning framework. However, most classification methods are designed to classify images into a fixed number of classes and are not designed for ranking images. Thus, the ranking results might be suboptimal.

This paper proposes to rank images through a framework that does not require distance calculation — a computationally expensive process. This is fundamentally different from most existing methods. The proposed framework is able to deal with retrieval tasks with one query as well as in relevance feedback. In contrast to this, most of the above-mentioned methods were designed to be used in relevance feedback only, e.g., `InstRank`, `Qsim` and `BALAS`.

It is also worth noting that [12] proposed a CBIR paradigm called query-by-semantic-example which maps and in turn retrieves images in a semantic space. Here a set of semantic-level concepts has to be predefined in order to construct the semantic features. On the contrary, the relevance features used in this paper are automatically generated — users do not need to specify them.

3. FRAMEWORK

Generally speaking, a CBIR system is composed of four parts [1]: (i) a given image database $\mathcal{D}$; (ii) a query $\mathcal{Q}$; (iii) a model $\mathbb{F}(\mathcal{Q}, \mathcal{D})$ to model the relationships between images in $\mathcal{Q}$ and $\mathcal{D}$; and (iv) a ranking scheme $R(\mathcal{D}|\mathcal{Q})$ which defines an ordering among the database images w.r.t. $\mathcal{Q}$. On the other hand, a ranking system consists of three components: (i) a given data set $\hat{\mathcal{D}}$; (ii) a model $\hat{\mathbb{F}}(\hat{\mathcal{D}})$ to model the relationships between instances in $\hat{\mathcal{D}}$; and (iii) a ranking scheme $\hat{R}(\hat{\mathcal{D}})$ which produces an ordering for all the instances in $\hat{\mathcal{D}}$.

Here, we propose to map the image database $\mathcal{D}$ from the original feature space $\mathbb{R}^d$ into a new space $\mathbb{R}^t$ to form a new database $\mathcal{D}'$ by using an ensemble of t ranking models, i.e., $\tilde{\mathbb{F}} = [\hat{\mathbb{F}}_1, \hat{\mathbb{F}}_2, \cdots, \hat{\mathbb{F}}_t]$. Given a query $\mathcal{Q}$, we first map it into the new space to obtain $\mathcal{Q}'$, and then we employ a ranking scheme $R'(\mathcal{D}'|\mathcal{Q}')$ to rank the images in $\mathcal{D}'$. We show in this paper that the ensemble of ranking models, i.e., $\tilde{\mathbb{F}}$, can be implemented using an anomaly detector called Isolation Forest, or iForest [10].

iForest builds an ensemble of iTrees to detect anomalies. An iTree is a random binary tree, constructed using a small sub-sample of the given data set. In fact, each iTree describes some profile underlying the data, and it is also a ranking model which produces an output in terms of path length for a test instance: a long (short) path length indicates that the instance is relevant (irrelevant) to the profile. Instances identified to be irrelevant to the various profiles modeled by a number of iTrees are deemed to be anomalies. Detailed algorithms of iTree and iForest are provided in Appendix.

In our framework, we build an iForest which is composed of t iTrees to map images from the original feature space to the relevance feature space, i.e., $\mathbb{R}^d \rightarrow \mathbb{R}^t$. Here different iTrees profile different aspects of the database. We treat each iTree as a feature descriptor, and the feature value (i.e., path length) is a measure of relevance w.r.t. the profile modeled by the iTree. For an image, the representation in the new space is a vector of relevance features; hence the name **relevance feature mapping**. We call the framework, incorporating the relevance feature mapping, `ReFeat`. Details of `ReFeat` are presented in the next section.

4. METHODOLOGY

4.1 Off-line Modeling

`ReFeat` has two stages. In the off-line modeling stage, we build an iForest for the given database $\mathcal{D}$. Here t iTrees are constructed, each built on a sub-sample of randomly selected ψ images from $\mathcal{D}$. For an image $\mathbf{x} \in \mathcal{D}$, we estimate its relevance feature value $\ell_i(\mathbf{x})$ on each iTree T_i ($i \in \{1, 2, \ldots, t\}$) and map $\mathbf{x}$ to the relevance feature space as: $\mathbf{x}' = [\ell_1(\mathbf{x}), \ell_2(\mathbf{x}), \cdots, \ell_t(\mathbf{x})]^T$. All the images in $\mathcal{D}$ are mapped through the relevance feature mapping to form a new database $\mathcal{D}'$. Note that this stage does not require any user intervention. Thus, $\mathcal{D}'$ is generated off-line to accelerate the following on-line retrieval process.

4.2 On-line Retrieval with One Query

Given a query image $\mathbf{q}$, ReFeat represents it as a vector of t relevance features: $\mathbf{q}' = [\ell_1(\mathbf{q}), \ell_2(\mathbf{q}), \cdots, \ell_t(\mathbf{q})]^T$, where $\ell_1(\mathbf{q}), \cdots, \ell_t(\mathbf{q})$ are obtained on-line. To retrieve the relevant images, we first assign a weight to each feature according to $\mathbf{q}$: a high weight is assigned to a feature which signifies that $\mathbf{q}$ is relevant to the profile modeled by the feature; otherwise, a low weight is assigned. Then the ranking score for every image in the database is computed using a weighted average of its relevance feature values. The images having the highest scores are regarded to be the most relevant to the query. To implement this, we define a weight for the ith feature as:

$$w_i(\mathbf{q}) = \frac{\ell_i(\mathbf{q})}{c(\psi)} - 1, \tag{1}$$

where $c(\psi)$ is a normalization term which is the average path length of a ψ-sized iTree, and it is defined as $c(\psi) = 2\big(ln(\psi-1) - (\psi-1)/\psi + E\big)$ [10] ($E \approx 0.5772$ is Euler's constant). Note that the minimum and maximum path lengths estimated by a ψ-sized iTree are 1 and $\psi - 1$, respectively; and thus the weight produced by Equation (1) has the range: $[\frac{1}{c(\psi)} - 1, \frac{\psi-1}{c(\psi)} - 1]$ (for example, $w_i(\mathbf{q}) \in [-0.6966, 1.1236]$ with $\psi = 8$).

Finally, the ranking score of an image $\mathbf{x}$ w.r.t. the query $\mathbf{q}$ is given by the average weighted feature value:

$$Score(\mathbf{x}|\mathbf{q}) = \frac{1}{t}\sum_{i=1}^{t}\big(w_i(\mathbf{q}) \times \ell_i(\mathbf{x})\big). \tag{2}$$

$Score(\mathbf{x}|\mathbf{q})$ can be negative. If necessary, positive scores can be produced by using an exponential mapping. For the rest of this paper, we refer to the ranking based on average weighted feature values as our ranking scheme.

It is worth noting that the off-line modeling of iForest utilizes no distance or similarity measure [10], and the proposed on-line ranking scheme also avoids distance or similarity calculation through Equations (1) and (2). This characteristic differentiates ReFeat from most existing methods which are based on certain distance or similarity measures.

An illustrative example of ReFeat is shown in Figure 1. The synthetic data set is composed of four classes, each contains 50 instances. Here Concept 1 to Concept 4 are used to represent four different semantic concepts in a CBIR task. The instances are randomly drawn from Gaussian distributions with unit standard deviation and means located at $(2, 2)$, $(-2, 2)$, $(-2, -2)$ and $(2, -2)$, respectively. The data is presented in Figure 1(a). We map the data into a relevance feature space using 1000 iTrees, each built on a 8-sized sub-sample. The contours of equally-weighted average feature values are shown in Figure 1(b). Then, given a query which is represented by the black diamond (⧫) in Figure 1(c), ReFeat produces ranking score for every instance in the data set according to Equation (2), and the resultant contours of the ranking scores are plotted in Figure 1(c). The query belongs to Concept 2, and most instances in Concept 2 are highly ranked by ReFeat, compared to instances from the other three concepts. This shows that the score calculated by the average weighted feature value is a proper relevance measure for ranking instances with respect to a given query.

4.3 On-line Retrieval in Relevance Feedback

If relevance feedbacks are available from the user, we use them to refine the retrieval result by modifying the feature weights. Here the query is denoted by $\mathcal{Q} = \mathcal{P} \cup \mathcal{N}$, where $\mathcal{P}$ is the set of images from positive feedbacks and the initial query, and $\mathcal{N}$ is the set of images from negative feedbacks. After the initial query, they are initialized as follows: $\mathcal{P} = \{\mathbf{q}\}$ and $\mathcal{N} = \varnothing$. Then, $\mathcal{P}$ and $\mathcal{N}$ are enriched with the images labeled by the user in the relevance feedback process. Here, we calculate the weight for the ith feature based on a positive feedback $\mathbf{z}^+ \in \mathcal{P}$ in the same way as that for the initial query, and produce the weights based on a negative feedback $\mathbf{z}^- \in \mathcal{N}$ through negation. Formally, we have:

$$w_i^+(\mathbf{z}^+) = \frac{\ell_i(\mathbf{z}^+)}{c(\psi)} - 1, \text{ and } w_i^-(\mathbf{z}^-) = 1 - \frac{\ell_i(\mathbf{z}^-)}{c(\psi)}.$$

Then the resultant weight for feature i due to $\mathcal{P}$ (or $\mathcal{N}$) is generated using an averaging scheme which is defined as follows (here $|\cdot|$ denotes the size of a set):

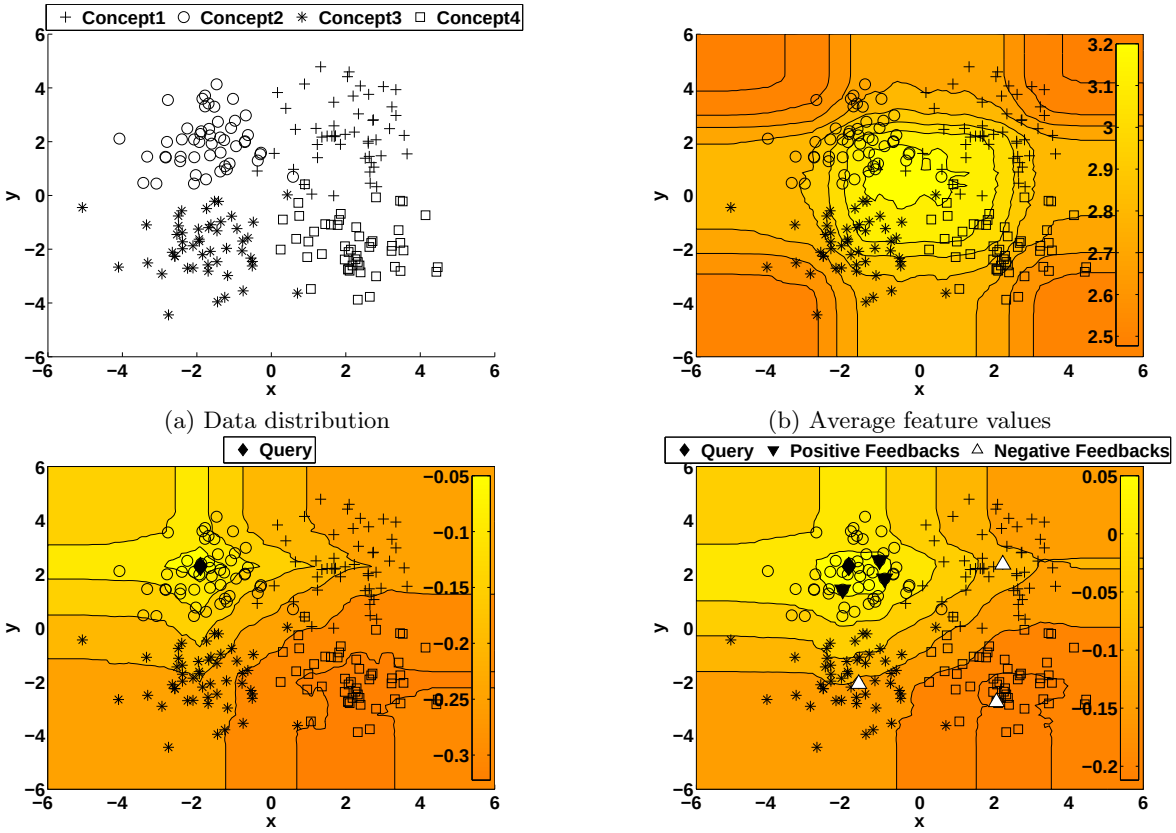
$$w_i^+(\mathcal{P}) = \frac{1}{|\mathcal{P}|}\sum_{k=1}^{|\mathcal{P}|} w_i^+(\mathbf{z}_k^+), \text{ and } w_i^-(\mathcal{N}) = \frac{1}{|\mathcal{N}|}\sum_{s=1}^{|\mathcal{N}|} w_i^-(\mathbf{z}_s^-).$$

Now the final weight for the ith feature can be obtained by aggregating $w_i^+(\mathcal{P})$ and $w_i^-(\mathcal{N})$. The aggregation can be realized in different ways. Here we use a simple summing method: $w_i(\mathcal{Q}) = w_i^+(\mathcal{P}) + \gamma w_i^-(\mathcal{N})$, where $\gamma \in (0, 1]$ is a trade-off parameter accounting for the relative weights of the contributions between positive and negative images. It is reasonable that positive images make more contribution to the final ranking than negative ones. Since the farther an image lies from positive images, the less likely that it is a relevant one. However, we can not draw an opposite conclusion for negative images: if an image lies far from negative images, the possibility that it is a relevant one is not necessary enhanced, since it may not be close to positive images either. Similar strategies were employed in previous works (e.g., [7, 20]).

Finally, ReFeat estimates the ranking score for every image in the database using Equation (2) (by replacing $w_i(\mathbf{q})$ with $w_i(\mathcal{Q})$), and returns the images by ranking them in a descending order according to their scores. Continue with our synthetic example where the retrieval result after the initial query is shown in Figure 1(c); with 3 positive feedbacks and 3 negative feedbacks, ReFeat refines the retrieval result by producing improved ranking scores, as shown in Figure 1(d).

4.4 Time and Space Complexities

We now analyze the time complexity of ReFeat. In the off-line modeling stage, building the iForest model takes $O\big(t\psi \log \psi\big)$ and the mapping from $\mathcal{D}$ to $\mathcal{D}'$ costs $O\big(|\mathcal{D}|t \log \psi\big)$ [10]. Thus, a total time complexity of $O\big((|\mathcal{D}| + \psi)t \log \psi\big)$ is required. In the on-line retrieval stage, the relevance feature mapping for the query costs $O\big(t \log \psi\big)$, calculating weights takes $O\big(|\mathcal{Q}|t\big)$, and producing relevance scores for all images in the database costs $O\big(|\mathcal{D}|t\big)$. Thus, for a query session, ReFeat has a time complexity of $O\big((|\mathcal{D}| + |\mathcal{Q}| + \log \psi)t\big)$. It is worth noting that $|\mathcal{Q}|$ is much smaller than $|\mathcal{D}|$, and both t and ψ are fixed at the beginning of the off-line modeling stage which are never changed in on-line retrieval. Thus, ReFeat has a linear time complexity with respect to $|\mathcal{D}|$ in



(a) Data distribution

(b) Average feature values

(c) Retrieval with one query

(d) Retrieval in relevance feedback

Figure 1: An illustrative example of ReFeat. Detailed descriptions are provided in Sections 4.2 and 4.3.

both off-line modeling and on-line retrieval stages[1], which makes it possible to scale up to large image databases. Table 1 lists the time complexities of ReFeat as well as three other methods. It shows that ReFeat has a relatively low time complexity in on-line retrieval although it needs an additional modeling stage. Note that we also compare BALAS and MRBIR in our experiments. Although it is difficult to analyze their complexities, the experimental results show that BALAS and MRBIR usually spend more time than ReFeat.

The space requirement of our off-line model is also linear with respect to $|\mathcal{D}|$, since the database $\mathcal{D}'$ costs $O(|\mathcal{D}|t)$ and iForest requires $O((2\psi-1)tb)$ memory space only [10], where b is the memory size of a node in iTrees.

5. EXPERIMENTS

The performance of ReFeat is evaluated on the following two aspects: (i) retrieval with one query, and (ii) retrieval in relevance feedback. Our experiments are conducted using the COREL image database [19] which contains 100 categories and each category has 100 images. Each image is represented by a 67-dimensional feature vector, which consists of 11 shape, 24 texture and 32 color features. To test the performance, we randomly select 5 images from each category to serve as the initial queries. For a query, the images within the same category are regarded as relevant and the rest are irrelevant. For each query, we continue to perform 5 rounds of relevance feedback. In each round, 2 positive and 2 negative feedbacks are provided. This relevance feedback process is also repeated 5 times with 5 different series of feedbacks. Finally, the average results with one query and in different feedback rounds are recorded. Note that there is no feature selection although it may be beneficial. The same features are used by all the compared methods because we are only interested in the relative instead of absolute performances of the methods. The experiments are conducted on a Pentium 4 machine with a 1.86 GHz CPU and 2 GB memory.

We employ a commonly used measure, i.e., PR-curve, to evaluate the retrieval performances with one query. However, in relevance feedback, a single PR-curve is not enough to reveal the performance changes with the increasing number of feedbacks. Thus, we use BEP-graph [20, 19] and effectiveness [3]. A BEP-graph is obtained by connecting the Break-Event-Points (BEPs) after different feedback rounds, where a BEP denotes the point in which both precision and recall have the same value. The higher the BEP value, the better the performance. Effectiveness η_S is a quantitative measure defined as:

$$\eta_S = \begin{cases} |\mathcal{R} \bigcap \mathcal{T}_S|/|\mathcal{R}| & \text{if } |\mathcal{R}| \leq S \\ |\mathcal{R} \bigcap \mathcal{T}_S|/|\mathcal{T}_S| & \text{if } |\mathcal{R}| > S \end{cases},$$

where S denotes the number of relevant images the user wants to retrieve, $\mathcal{R}$ represents the set of relevant images in the database w.r.t. the query, and $\mathcal{T}_S$ is the set of the top

[1]Here we omit the linear time complexity for finding the highest ranking scores as it is inevitable for most methods.

Table 1: Time complexities of ReFeat, Euclidean, InstRank [6] and Qsim [20]. Here d is the dimension of the database $\mathcal{D}$ and "–" means not required. InstRank and Qsim are methods dealing with relevance feedback only.

	Off-line modeling	On-line retrieval	
		with one query	in relevance feedback
ReFeat	$O((\lvert\mathcal{D}\rvert + \psi) \times t \times \log \psi)$	$O((\lvert\mathcal{D}\rvert + \log \psi) \times t)$	$O((\lvert\mathcal{D}\rvert + \lvert\mathcal{Q}\rvert) \times t)$
Euclidean	–	$O(\lvert\mathcal{D}\rvert \times d)$	$O(\lvert\mathcal{D}\rvert \times \lvert\mathcal{Q}\rvert \times d)$
InstRank	–	N/A	$O(\lvert\mathcal{D}\rvert \times \lvert\mathcal{Q}\rvert \times d)$
Qsim	–	N/A	$O(\lvert\mathcal{D}\rvert \times \lvert\mathcal{Q}\rvert \times (d + \lvert\mathcal{P}\rvert))$

S ranked images returned by the system. The bigger the value of η_S, the better the performance. As in [20], we set $S = 200$.

The efficacy and efficiency of ReFeat are validated in the next subsection, followed by empirical studies showing the effectiveness of the relevance feature mapping, the influence of increasing database dimension, and the effect of different parameter settings in ReFeat.

5.1 Comparison with Existing Methods

As a new ranking framework for CBIR, ReFeat is first compared with Euclidean method and MRBIR [7] when no relevance feedback is performed. In relevance feedback, Qsim [20], InstRank [6], MRBIR [7] and BALAS [17] are employed for benchmarking. Note that Qsim, InstRank and BALAS are designed to be used in relevance feedback only, and we use the Euclidean method as their base method before relevance feedback kicks in. The three parameters in ReFeat have the following default settings: number of relevance features $t = 1000$, sub-sample size $\psi = 8$ and trade-off parameter $\gamma = 0.25$. The default parameter settings are used for all the other compared methods.

The PR-curves of ReFeat, Euclidean and MRBIR for retrieval with one query are presented in Figure 2, which show that ReFeat outperforms the other two methods. We also conduct t-tests to gain further insight. For each of the 500 queries, we calculate the BEP and effectiveness values using every compared methods. The win/draw/loss counts are shown in Table 2, where a win/loss is counted if ReFeat performs better/worse on a query than the comparing approach, otherwise a draw is counted. Then a paired t-test at 5% significance level is performed for the recorded BEP (and effectiveness) series. The results are presented in Table 2 which show that ReFeat significantly outperforms Euclidean and MRBIR, in terms of both BEP and effectiveness.

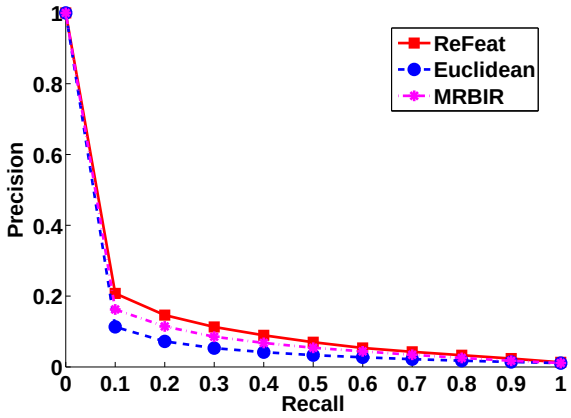


Figure 2: PR-curves of the compared methods.

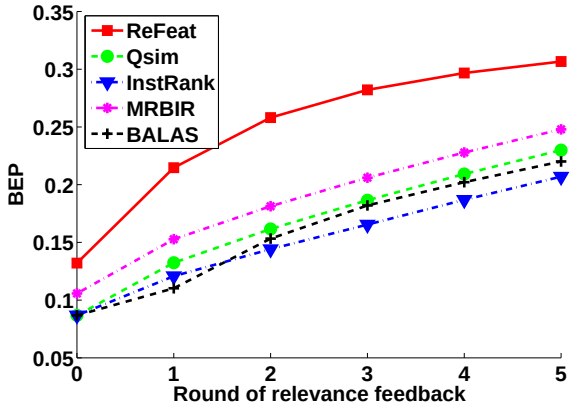


Figure 3: PR-curves and BEP-graphs of the compared methods.

Table 2: Win/draw/loss counts and t-test results of ReFeat against Euclidean and MRBIR for retrieval with one query. The number in bracket "[ ]" is the probability of rejecting the hypothesis that ReFeat is significant better than the compared method.

	BEP	Effectiveness
Euclidean	345/0/155	333/36/131
	$[5.7 \times 10^{-27}]$	$[2.0 \times 10^{-28}]$
MRBIR	328/0/172	331/34/135
	$[4.5 \times 10^{-9}]$	$[8.4 \times 10^{-15}]$

As for retrieval in relevance feedback, the BEP-graphs are shown in Figure 3, and the effectiveness values are tabulated in Table 3 where the best performance at each round has been boldfaced. Note that Round0 presents the retrieval performances with one query only, where Euclidean is used as the base method for Qsim, InstRank and BALAS. The results clearly show that ReFeat achieves the best BEP and effectiveness no matter how many feedbacks are provided. Since ReFeat has a superior performance with both one query and relevance feedbacks, we can conclude that ReFeat is effective for CBIR.

Apart from the retrieval performance, processing time is also an important factor in CBIR. The average on-line pro-

Table 3: Average effectiveness $\bar{\eta}_{200}$ (%) values of ReFeat (RF), Qsim (QS), InstRank (IR), MRBIR (MR) and BALAS (BA).

	RF	QS	IR	MR	BA
Round0	**18.78**	12.52	12.52	14.25	12.52
Round1	**29.14**	18.46	16.69	20.15	14.55
Round2	**34.42**	21.87	19.41	23.48	19.64
Round3	**37.47**	24.67	21.79	26.20	23.10
Round4	**39.20**	27.11	24.08	28.57	25.28
Round5	**40.60**	29.33	26.19	30.66	27.40

cessing time of all compared methods is presented in Table 4 where the shortest time cost at each round has been boldfaced. The results show that ReFeat has the best efficiency except that it spends a bit more time than Euclidean at Round0. Notice that ReFeat achieves the shortest and near constant processing time regardless of the feedback rounds. The time is independent of the number of feedbacks because the time complexity of ReFeat for retrieval in relevance feedback is dominated by $O(|\mathcal{D}|\times t)$ as $|\mathcal{Q}| \ll |\mathcal{D}|$. InstRank also has a near constant time cost because the distances calculated in previous feedback rounds are saved for the following rounds. MRBIR has to iteratively update the ranking result with expensive large matrix operations.

Although ReFeat has an off-line modeling stage, it costs 2.87 seconds only for our database containing 10000 images. We believe that it pays to employ such an off-line modeling stage because of the good performance and quick processing time achieved by ReFeat in on-line retrieval.

Table 4: Average processing time (in *ms*) of ReFeat (RF), Qsim (QS), InstRank (IR), MRBIR (MR) and BALAS (BA).

	RF	QS	IR	MR	BA
Round0	27.8	**27.2**	**27.2**	701.0	**27.2**
Round1	**27.1**	72.4	32.5	1353.5	257.3
Round2	**27.4**	150.6	33.4	1354.1	314.8
Round3	**27.5**	270.6	34.2	1353.8	374.9
Round4	**27.7**	431.8	34.8	1353.8	445.4
Round5	**27.8**	637.7	35.5	1353.9	516.4

5.2 Relevance Feature Mapping

Recall that ReFeat is a two-stage process, where the first maps images in a database to a relevance feature space, and the second ranks the images in this space. Experiments in the last subsection have already shown that ReFeat outperforms the other methods which use the original feature space. Here, we hypothesize that the performance of existing ranking methods can be improved using our relevance features. Thus, we perform Qsim and InstRank in the relevance feature space (denoted by Qsim-ReFeat and InstRank-ReFeat, respectively), and the results are shown in Figure 4. Exactly the same relevance feature mapping is employed for all methods that use it. MRBIR and BALAS are omitted in this experiments due to their high computational costs. Note that the same conclusion can always be made

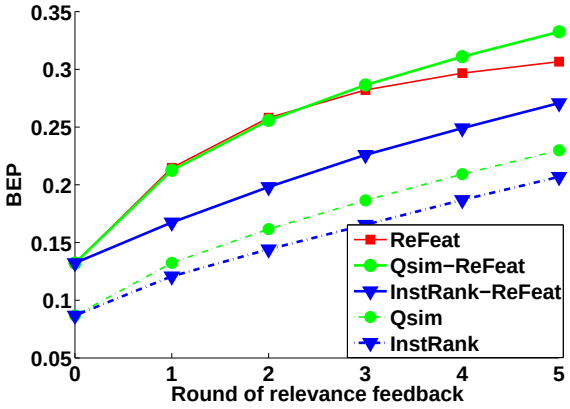


Figure 4: BEP-graphs of ReFeat, Qsim-ReFeat, InstRank-ReFeat, Qsim and InstRank.

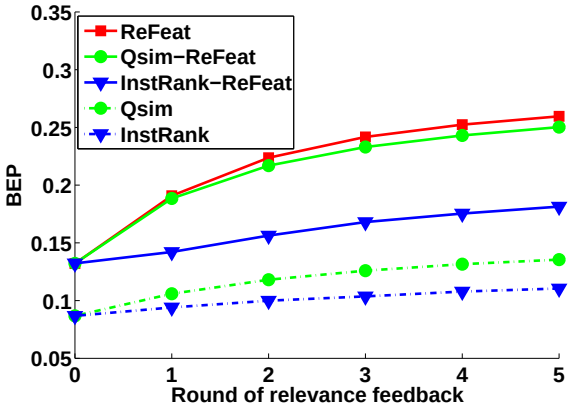


Figure 5: BEP-graphs of ReFeat, Qsim-ReFeat, InstRank-ReFeat, Qsim and InstRank. Here, the performance is evaluated on the ranking of non-feedbacks.

for both BEP and effectiveness. Thus, we only provide the BEP results hereafter.

As shown in Figure 4, with the help of the relevance feature mapping, Qsim-ReFeat and InstRank-ReFeat significantly outperform their original versions, i.e., Qsim and InstRank. These observations clearly show that our relevance feature space is more suitable for ranking than the original space. Thus, we can conclude that the power of ReFeat comes from the relevance feature mapping. In addition, ReFeat is comparable with Qsim-ReFeat except that Qsim-ReFeat achieves the best performance at the last two feedback rounds.

Note that our computation of retrieval performance has included feedback images, following the same convention as previously reported results (e.g., [6, 20]). However, this calculation not only inflates the performance by including feedbacks, but also has an unfair advantage over methods which do not use distance measures — that may not return zero distances for feedbacks. Thus, we have re-calculated the retrieval performance by excluding the feedbacks, and the results are presented in Figure 5 — the "true" retrieval performances. It shows that our ranking scheme outperforms the others at ranking non-feedbacks, and paired t-tests on

the BEP values also indicate significant improvement at each feedback round (details are omitted due to space limitation). These results show that ReFeat is good at retrieving new relevant images rather than returning the previously labeled feedbacks.

Moreover, the processing time reported in Table 5 shows that ReFeat has the best efficiency among the three methods applied in the relevance feature space. This is because ReFeat is designed to be used in this space by taking advantages of the relevance features.

Table 5: Average processing time (in *ms*) of ReFeat (RF), Qsim-ReFeat (QS-RF), InstRank-ReFeat (IR-RF), Qsim (QS) and InstRank (IR).

	RF	QS-RF	IR-RF	QS	IR
Round0	27.8	347.6	347.6	**27.2**	**27.2**
Round1	**27.1**	460.6	421.6	72.4	32.5
Round2	**27.4**	537.3	422.5	150.6	33.4
Round3	**27.5**	654.9	423.3	270.6	34.2
Round4	**27.7**	812.6	424.0	431.8	34.8
Round5	**27.8**	1013.8	424.6	637.7	35.5

5.3 Increasing Dimensions

Here we investigate the effect on the retrieval performance and processing time as a result of an increase in database dimensions. Recall that every image in the COREL database is represented by a 67-dimensional feature vector containing shape, texture and color features. Here we denote the database as $\mathcal{D}_{[S,T,C]}$, and construct three other databases: (i) $\mathcal{D}_{[S]}$ employs 11 shape features only; (ii) $\mathcal{D}_{[S,T]}$ uses 35 features, consisting of 11 shape and 24 texture features; and (iii) $\mathcal{D}_{[S,T,C,R]}$ is a 200-dimensional database, created by adding 133 random features to $\mathcal{D}_{[S,T,C]}$ (each random feature is generated from a uniform distribution). All the compared methods are evaluated on the four databases, and the retrieval results with one query and in feedback Round5 are shown in Figure 6, Figure 7 and Table 6 (here the results at other feedback rounds have been omitted for compactness).

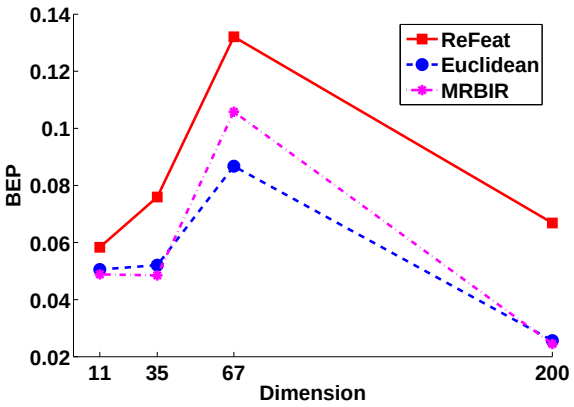


Figure 6: BEP values of the compared methods for retrieval with one query evaluated on the four databases.

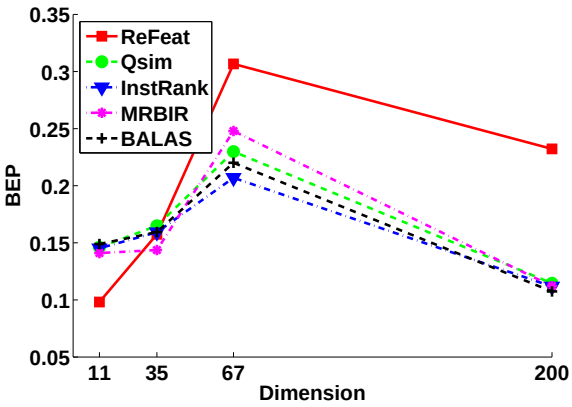


Figure 7: BEP values of the compared methods in relevance feedback Round5 evaluated on the four databases.

Figures 6 and 7 show that every method achieves its best performance on $\mathcal{D}_{[S,T,C]}$ out of the four databases. On the database $\mathcal{D}_{[S,T,C,R]}$ with randomly generated features, ReFeat outperforms the other methods with the lowest performance degradation, when compared with that on $\mathcal{D}_{[S,T,C]}$. For example, at Round5, ReFeat only degrades 24%, which is much better than 50% for Qsim, 46% for InstRank, 54% for MRBIR and 51% for BALAS. These observations indicate that ReFeat has a good tolerance for random features.

Table 6 shows that Euclidean and InstRank spend the shortest time in low-dimensional cases, but their processing time increases linearly with respect to the database dimension. Qsim, MRBIR and BALAS spend much more time than ReFeat. ReFeat achieves constant time w.r.t. the database dimension, either dealing with one query or handling feedbacks. This enables our framework to scale up to high-dimensional databases without increasing the processing time.

5.4 Parameter Settings

This subsection studies the effect of the three parameters in ReFeat, i.e., sub-sample size ψ, number of relevance features t and trade-off parameter γ.

To gain an insight into the setting of ψ, we measure the diversity of iTrees in an iForest using the Shannon index [9]. It is a statistic for measuring the biodiversity of an ecosystem. The index is increased when having additional unique species or a greater species evenness. A bigger Shannon index indicates a larger diversity. In our experiments, each image is treated as an ecosystem, and different feature values represent different species in the ecosystem. The Shannon index is calculated for each image, and the final diversity of iTrees in an iForest is obtained by averaging the Shannon indices over all images. We set $\psi = 2^2, 2^3, \cdots, 2^{12}$ (with $t = 1000$ and $\gamma = 0.25$), and the resultant diversities are plotted in Figure 8. It is interesting to note that the diversity decreases when $\psi > 64$, although the number of possible feature values (i.e., possible species) increases. The BEP values of ReFeat are also shown in Figure 8. Since the best performance of ReFeat is obtained with $\psi = 8$ for our database and there is no benefit to use a large ψ (i.e., $\psi > 64$), we suspect that the optimal setting for any CBIR task is somewhere between the smallest ψ ($= 4$) and the

Table 6: Average processing time (in ms) of the compared methods tested on the four databases. Here, ReFeat, Euclidean, MRBIR, Qsim, InstRank and BALAS are denoted by RF, EU, MR, QS, IR and BA, respectively.

Database	Dimension	One query			Round5				
		RF	EU	MR	RF	QS	IR	MR	BA
$\mathcal{D}_{[S]}$	11	27.8	**6.8**	679.7	27.8	612.8	**10.6**	1353.9	332.2
$\mathcal{D}_{[S,T]}$	35	27.8	**16.1**	687.7	27.8	624.7	**22.5**	1353.9	410.5
$\mathcal{D}_{[S,T,C]}$	67	27.8	**27.2**	701.0	**27.8**	637.7	35.5	1353.9	516.4
$\mathcal{D}_{[S,T,C,R]}$	200	**27.8**	72.0	733.3	**27.8**	691.8	89.6	1353.9	848.1

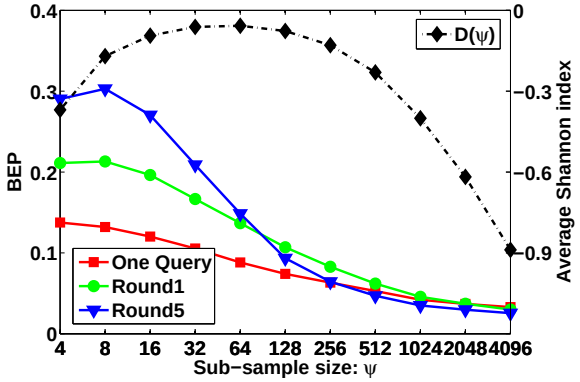


Figure 8: The effect of different ψ in ReFeat. Here the results are presented in terms of BEP values for retrieval with one query and in feedback Round1 and Round5. $D(\psi)$ is the average Shannon index calculated for the iTrees in an iForest.

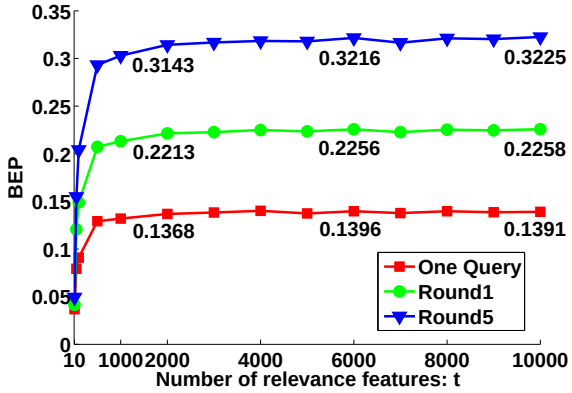


Figure 9: The effect of different t in ReFeat. Here the results are presented in terms of BEP values for retrieval with one query and in feedback Round1 and Round5.

diversity peak. This can be used as an empirical guideline for setting the sub-sample size.

Figure 9 shows the BEP values of ReFeat as the number of relevance features t varies from 10, 50, 100, 500, 1000, 2000, 3000, $\cdots$, 9000, to 10000 (with $\psi = 8$ and $\gamma = 0.25$). It shows that the retrieval performance of ReFeat rapidly increases with the increase of t when t is relatively small. Even with a sufficiently large t, the performance still appears to rise without overfitting. These observations make it possible to improve the performance of ReFeat by adding more relevance features. However, when selecting t, the trade-off between performance and processing time should be considered.

We also study how γ affects the performance of ReFeat in relevance feedback process. The results are shown in Figure 10. ReFeat achieves good performance when $\gamma \in [0.1, 0.4]$, signifying that negative images make less contribution than positive ones.

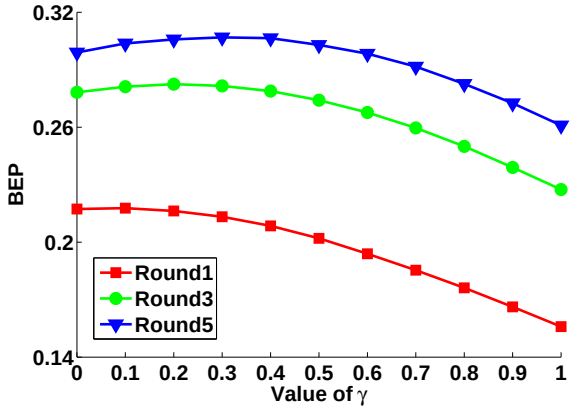


Figure 10: The effect of different γ in ReFeat. Here the results are presented in terms of BEP values for retrieval in feedback Round1, Round3 and Round5.

6. DISCUSSION

For a successful application in the proposed framework, the necessary characteristics of ranking models are: (i) each individual model provides a ranking of images through some profile underlying the database; and (ii) each model is generated efficiently so that the multiple models, representing multiple profiles of the database, can be generated very quickly to form the relevance feature space. We show in this paper that iTrees work well in our framework, but we are not aware of any other existing anomaly detection models[2] that satisfy the above characteristics. Most of the models such as distance-based, density-based or SVM anomaly detection methods either build a global model (thus cannot represent different profiles of the database) or have high computational cost in terms of time and space complexities [2]. Whether

[2]An anomaly detection model is basically a ranking model which attempts to rank the most anomalous instances at the top to the most normal instances at the bottom.

any other ranking model satisfies these characteristics is an open problem.

7. CONCLUSIONS

This paper proposes a novel ranking framework for content-based image retrieval with a unique relevance feature mapping obtained from an ensemble of ranking models. We employ an ensemble of iTrees to transform from the original feature space to the proposed new relevance feature space. We show that the new relevance feature space is more suitable than the original one for ranking database images with respect to a given query as well as subsequent feedbacks. We also show that the relevance feature space accounts for the significant performance improvement of several existing ranking methods when compared to those applied in the original feature space. Moreover, our proposed new ranking scheme is better than the others when they are evaluated in the same footing, in terms of both retrieval performance and time complexity.

The proposed framework has the following unique characteristics: (i) it utilizes no distance or similarity measure and has linear time and space complexities w.r.t. the database size when building its model and mapping the database off-line; (ii) it has constant on-line retrieval time, irrespective of the relevance feedback rounds; (iii) it can deal with high-dimensional image databases with constant time complexity; and (iv) it has a good tolerance for random features.

Theoretical and intuitive reasons underpinning the relevance feature mapping will be developed in the near future. We will also study the applications of `ReFeat` on other types of retrieval tasks.

Acknowledgements

This work was completed while Guang-Tong was visiting Monash University partly supported by a scholarship from Shandong University. Zhi-hua Zhou has given Guang-Tong a strong foundation in CBIR when he visited Nanjing University before this project. Zhouyu Fu has provided many helpful comments in the early draft. Suggestions from the anonymous reviewers have also helped to improve the clarity of this paper.

8. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley Longman, Boston, MA, 1999.

[2] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–58, 2009.

[3] G. Ciocca and R. Schettini. A relevance feedback mechanism for content-based image retrieval. *Information Processing and Management*, 35(5):605–632, 1999.

[4] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–60, 2008.

[5] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007.

[6] G. Giacinto and F. Roli. Instance-based relevance feedback for image retrieval. In *Advances in Neural Information Processing Systems 17*, pages 489–496, Vancouver, Canada, 2005.

[7] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang. Manifold-ranking based image retrieval. In *Proceedings of the 12th ACM International Conference on Multimedia*, pages 9–16, New York, 2004.

[8] X. He, W.-Y. Ma, and H. Zhang. Learning an image manifold for retrieval. In *Proceedings of the 12th ACM International Conference on Multimedia*, pages 17–23, New York, 2004.

[9] C. J. Krebs. *Ecological Methodology*. HarperCollins, New York, 1989.

[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 413–422, Pisa, Italy, 2008.

[11] N. Panda and E. Y. Chang. Efficient top-k hyperplane query processing for multimedia information retrieval. In *Proceedings of the 14th ACM International Conference on Multimedia*, pages 317–326, Santa Barbara, CA, 2006.

[12] N. Rasiwasia, P. J. Moreno, and N. Vasconcelos. Bridging the gap: Query by semantic example. *IEEE Transactions on Multimedia*, 9(5):923–938, 2007.

[13] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in mars. In *Proceedings the 1997 International Conference on Image Processing*, pages 815–818, Washington, DC, 1997.

[14] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[15] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[16] G. Wu, E. Y. Chang, and N. Panda. Formulating context-dependent similarity functions. In *Proceedings of the 13rd ACM International Conference on Multimedia*, pages 725–734, Singapore, 2005.

[17] R. Zhang and Z. M. Zhang. Balas: Empirical bayesian learning in the relevance feedback for image retrieval. *Image and Vision Computing*, 24(3):211–223, 2006.

[18] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, 2003.

[19] Z.-H. Zhou, K.-J. Chen, and H.-B. Dai. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Transactions on Information Systems*, 24(2):219–244, 2006.

[20] Z.-H. Zhou and H.-B. Dai. Query-sensitive similarity measure for content-based image retrieval. In *Proceedings of the 6th IEEE International Conference on Data Mining*, pages 1211–1215, Hong Kong, China, 2006.

APPENDIX

This section details the methodology of iForest [10][3], which employs a two-stage process to detect anomalies. We provide some insights on how each iTree measures the relevance of instances with respect to a profile underlying the data. It helps to understand the relevance feature space.

In the first stage, iForest builds a collection of iTrees using random sub-samples of a data set. Each iTree is constructed by recursively random-partitioning a sub-sample along axis-parallel coordinates until instances are isolated. Here, an isolated instance is an instance that is being separated from the rest of instances by these random partitions. Details of this process can be found in Algorithms 1 and 2. Note that an iTree models a profile of the given random sub-sample, and different iTrees describe different profiles due to the randomness incurred in both the sub-sampling process and the tree building process.

Algorithm 1 : iForest($\mathcal{X}, t, \psi$)

Inputs: $\mathcal{X}$ - input data, t - number of iTrees, ψ - sub-sample size
Output: a set of t iTrees

Initialize Forest
for $i = 1$ to t **do**
  $X \leftarrow sample(\mathcal{X}, \psi)$
  Forest $\leftarrow$ Forest $\cup$ iTree(X)
end for
return Forest

Algorithm 2 : iTree(X)

Inputs: X - input data
Output: an iTree

if $|X| \leq 1$ **then**
  return $exNode\{Size \leftarrow |X|\}$
else
  let A be a list of attributes in X
  randomly select an attribute $a \in A$
  randomly select a split point p from max and min values of attribute a in X
  $X_l \leftarrow filter(X, a < p)$
  $X_r \leftarrow filter(X, a \geq p)$
  return $inNode\{Left \leftarrow$ iTree(X_l),
    $Right \leftarrow$ iTree(X_r),
    $SplitAtt \leftarrow a$,
    $SplitValue \leftarrow p\}$
end if

In the second stage, iForest calculates an anomaly score for each test instance based on its average path length. A path length is derived by passing the instances through an iTree until a specified height limit is reached. In our experiments, we set the height limit $h = ceiling(\log_2 \psi)$ as in [10]. This process is given by Algorithm 3. Here, a short path length means that we can easily isolate the instance from the majority of instances within a few random partitions. That is to say, instances having short path lengths always differ from the majorities on some characteristics. Recall that an iTree describes a data profile from a given sub-sample. Thus, instances having short path length are considered to be irrelevant to this data profile by having very different characteristics from the given sub-sample. On the other hand, instances with long path length are relevant to the profile by having similar characteristics. So the path length stipulated by an iTree actually measures the relevance of an instance with respect to the profile modeled by this iTree: a short (long) path length indicates that the instance is irrelevant (relevant) to the profile. Under the view of anomaly detection, instances identified to be irrelevant to the various profiles modeled by a number of iTrees are deemed to be anomalies, and instances relevant to the various profiles are normal points.

[3]Software download at
`http://sourceforge.net/projects/iforest/`.

Algorithm 3 : PathLength(x, T, e)

Inputs: $\mathbf{x}$ - an instance, T - an iTree, e - current path length; to be initialized to zero when first called
Output: path length of $\mathbf{x}$

if $e \geq h$ (height limit) or T is an external node **then**
  return $e + c(T.Size)$
  $\{c(m) = 2\big(ln(m-1) - (m-1)/m + E\big)$ ($E \approx 0.5772$ is Euler's constant)$\}$
end if
$a \leftarrow T.SplitAtt$
if $T.SplitValue \leq \mathbf{x}_a$ **then**
  return PathLength($\mathbf{x}, T.Right, e + 1$)
else if $x_a < T.SplitValue$ **then**
  return PathLength($\mathbf{x}, T.Left, e + 1$)
end if