Exploration-Exploitation of Eye Movement Enriched Multiple Feature Spaces for Content-Based Image Retrieval

Zakria Hussain[1], Alex P. Leung[2], Kitsuchart Pasupa[3], David R. Hardoon[4], Peter Auer[2], and John Shawe-Taylor[1]

[1] University College London, UK
{z.hussain,jst}@cs.ucl.ac.uk
[2] University of Leoben, Austria
{auer,alex.leung}@unileoben.ac.at
[3] University of Southampton, UK
k.pasupa@gmail.com
[4] Institute for Infocomm Research (I^2R), Singapore
drhardoon@i2r.a-star.edu.sg

Abstract. In content-based image retrieval (CBIR) with relevance feedback we would like to retrieve relevant images based on their content features and the feedback given by users. In this paper we view CBIR as an Exploration-Exploitation problem and apply a kernel version of the LinRel algorithm to solve it. By using multiple feature extraction methods and utilising the feedback given by users, we adopt a strategy of multiple kernel learning to find a relevant feature space for the kernel LinRel algorithm. We call this algorithm LinRelMKL. Furthermore, when we have access to eye movement data of users viewing images we can enrich our (multiple) feature spaces by using a tensor kernel SVM. When learning in this enriched space we show that we can significantly improve the search results over the LinRel and LinRelMKL algorithms. Our results suggest that the use of exploration-exploitation with multiple feature spaces is an efficient way of constructing CBIR systems, and that when eye movement features are available, they should be used to help improve CBIR.

Keywords: content-based image retrieval, LinRel, images, eye movements, multiple kernel learning, tensor kernel SVM

1 Introduction

Assume we have a database $\mathcal{D}$ of images and would like to find an image $x \in \mathcal{D}$ without having to conduct an exhaustive search of $\mathcal{D}$. If we have not tagged the images in the database, then we would need to retrieve images using some content-based image retrieval (CBIR) system utilising relevance-feedback [9]; a

tool designed for accessing *relevant* images from a database using their *content* such as colour, shape, texture etc., and receiving *feedback* from the user on the relevance of images presented so far. We will refer to content-based image retrieval using relevance-feedback as CBIR throughout this paper. Using this relevance information the CBIR system should find relevant images quickly. The goal of the CBIR system is to produce the target image in the least number of presented images.

In this paper, we view the underlying mechanics of a CBIR system as an exploration-exploitation problem. We apply the LinRel algorithm [3], a linear regression algorithm that efficiently carries out exploration-exploitation. LinRel makes use of side information – typically various image features – to estimate the relevance of an image. Since these estimates are imperfect, when selecting possibly relevant images the algorithm needs to trade-off between high estimated relevance and possible gain of information to improve the estimates.

Each image can be represented using many different feature extraction methods such as SIFT, histogram, colour, *etc.*, and so its not clear which ones to use for the LinRel algorithm – as some may be good for some search tasks but not for others. A recent line of research, called multiple kernel learning (MKL) aims at finding a good linear combination of feature spaces [15]. The idea is to find the best weighting of feature spaces for a given classification task. By viewing the relevance feedback as our classification outputs we can learn a combination of feature spaces using MKL, and present this new kernel to the kernelized version of the LinRel algorithm. This removes the need of choosing explicitly which feature spaces to use and also tackles the problem of combining feature spaces with a non-uniform combination, unlike [8]. This algorithm will be called LinRelMKL.

Finally, we will also assume that our CBIR system has access to an eye tracking device, allowing us to record users' eye movements whilst they view images.[5] This gives us an *extra set of features* for each image *viewed*. However, we do not have access to the eye movement features for the images not shown by the CBIR system. Therefore, we use a recent approach [11] of mapping the combination of image features we have derived (using MKL) into the eye movement feature spaces, using the tensor kernel support vector machine [12]. When used with LinRel alone we call this algorithm LinRelTensor and when also combined with MKL we call it LinRelMKLTensor.

We start with our problem definition in Section 2, followed by the LinRel, MKL, and tensor learning algorithms in Section 3, 4 and 5, respectively. Section 6 describes our final CBIR system. The experiments are described in Section 7, with concluding remarks in Section 8. We begin with some preliminary definitions.

Notation: Let $\mathbf{x} \in \mathbb{R}^m$ be an m-dimensional row vector, with $\mathbf{x}^\top$ denoting its transpose. Let $\phi : \mathbf{x} \to \mathcal{H} \subset \mathbb{R}^M$ be a mapping into a high dimensional feature space $\mathcal{H}$. A kernel function will be defined as $\kappa(I, I) = \langle \phi(\mathbf{x}_I), \phi(\mathbf{x}_I) \rangle$

[5] We will show later that having access to eye movement features can improve search results.

and accessed using an index set I. The cardinality of I will be made clear from context. We denote $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)^\top$ as the matrix containing t row vectors and $\mathbf{I}$ as the identity matrix. The covariance matrix can be written as $\mathbf{X}^\top\mathbf{X}$, with its inverse as $\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}$. The notation $\|\mathbf{x}\|_p$ will denote the p-norm of vector $\mathbf{x}$, where $p = 1, 2$.

2 Problem definition

In content-based image retrieval (CBIR) with relevance feedback [24, 8, 19, 18], one can consider a model motivated by a filtering task, where a user wants to find a set of images relevant to his query. When presented with a collage of images, the user marks[6] all images which are relevant to his query. The goal of the search algorithm is to present as many relevant images as possible in early iterations of the search.

Our work focuses on algorithms and experiments for this model. The online algorithms build upon the LinRel algorithm proposed in [3, Section 4]. Before discussing the LinRel algorithm we will describe the user model we consider, the model for relevance scores and also the different image feature extraction methods used throughout.

2.1 The user model for the filtering task

We assume that the user is looking for a set of relevant images I in an image database $\mathcal{D}$. The relevance of an image is determined by the user query, about which the search engine is informed only by relevance feedback. The goal of the search engine is to present mostly relevant images to the user, and only a small number of irrelevant images. The feedback of the user is given by a relevance score $y \in [0, 1]$, with 0 meaning not relevant, 1 meaning relevant, and possible degrees of relevance for values in between. The relevance score can be given by an explicit binary feedback (e.g. mouse clicks), or implicitly (e.g. by recorded eye movements). The formal protocol for this model is outlined in Fig. 1.

- In each iteration $t = 1, 2, \ldots$
  - The search engine selects an image $I_t \in \mathcal{D}$ and presents it to the user.
  - The user's feedback is given by the relevance score $y_t \in \{0, 1\}$.

Fig. 1: Original protocol for LinRel

The performance of the search engine is determined by the number of relevant images returned in a certain number of iterations. We note that in the protocol

[6] This will be done explicitly using mouse clicks. An implicit score of relevance could be found using eye movements. However we will not address this extension in the current paper.

of Figure 1 only a *single* image is presented in each round and the relevance score is *binary*. This is the original protocol used to develop the LinRel algorithm in [3]. In a realistic CBIR setting, the protocol should be extended to Fig. 2.

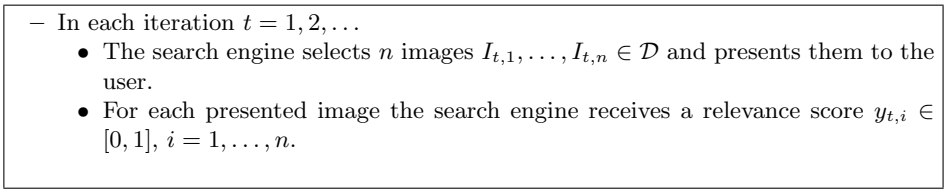

Fig. 2: A CBIR protocol for LinRel

Thus the search engine described in Figure 2 needs to select a fixed number n of images, with the relevance scores ranging between 0 and 1.

2.2 Modelling the relevance scores

To be able to learn about the user's query from the relevance scores, we are making assumptions about how the relevance scores are generated. We assume that an image I is represented by a normalised vector $\mathbf{x}_I \in \mathbb{R}^d$ of features (see subsection 2.3), with $\|\mathbf{x}_I\|_2 = 1$. Furthermore, we assume that the relevance score y_I of an image I is a random variable with expected value $\mathbb{E}[y_I] = \mathbf{x}_I \cdot \mathbf{w}$, $\mathbf{w} \in \mathbb{R}^m$, such that the expected relevance score is a linear function of the image features. The unknown weight vector $\mathbf{w}$ is essentially the representation of the user's query and determines the relevance of images.

2.3 Features extracted from images

The feature extraction methods can be found below in Table 1, with a detailed description of each method given in [14]. For each image I we will assume that each of the 11 feature extraction methods of Table 1 has been carried out. Hence, for each image I we will have feature vectors $\mathbf{x}_{I,1}, \ldots, \mathbf{x}_{I,11}$.

A typical method of using all of these feature vectors would be to concatenate them together and construct a single kernel matrix, and run this through a kernelized CBIR system [8]. This would correspond to a uniform weighting of the feature vectors (*i.e.*, kernels). However, it may be the case for some search tasks that, for instance *"Histogram of four Sobel edge direction"* would be more important than *"CIE L*a*b* colour of two dominant colour clusters"* features. Using a uniform weighting only assumes that both features are equally important. Therefore we will apply kernels for each of the feature extraction methods outlined in Table 1, and apply MKL to find a linear combination of these feature spaces (*i.e.*, kernels). For certain search tasks, this linear combination results in a non-uniform weighting, giving rise to higher weightings for more useful feature extraction methods.

Feature	dimensions
DCT coefficients of average colour in rectangular grid	12
CIE L*a*b* colour of two dominant colour clusters	6
Histogram of local edge statistics	80
Haar transform of quantised HSV colour histogram	256
Histogram of interest point SIFT features	256
Average CIE L*a*b* colour	15
Three central moments of CIE L*a*b* colour distribution	45
Histogram of four Sobel edge directions	20
Co-occurrence matrix of four Sobel edge directions	80
Magnitude of the 16×16 FFT of Sobel edge image	128
Histogram of relative brightness of neighbouring pixels	40

Table 1. Visual features extracted from images

3 The LinRel algorithm and its "kernelized" counterpart

In [3] the LinRel algorithm was devised for a slight variant of the model described in the previous section. In this section we describe the LinRel algorithm with the necessary modifications to accommodate our model. We restrict ourselves to the case $n = 1$, *i.e.*, only one image is presented to the user in each iteration. The general case *i.e.*, image collages, will be discussed later.

The user model for the filtering tasks (in particular the model for the relevance scores) confronts the search engine with an exploration-exploitation tradeoff. Typically the search engine will maintain an implicit or explicit representation of an estimate $\hat{\mathbf{w}}$ of the unknown weight vector $\mathbf{w}$. When selecting the next image for presentation to the user, the search engine might simply select the image with highest estimated relevance score based on $\hat{\mathbf{w}}$. But since the estimate $\hat{\mathbf{w}}$ might be inaccurate, this exploitative choice might be suboptimal. Alternatively, the search engine might exploratively select an image for which the user feedback improves the accuracy of the estimate $\hat{\mathbf{w}}$, enabling better image selections in subsequent iterations.

In each iteration t, LinRel obtains an estimate $\hat{\mathbf{w}}_t$ by solving the linear regression problem

$$\mathbf{y}_t \approx \mathbf{X}_t \cdot \hat{\mathbf{w}}_t,$$

where

$$\mathbf{y}_t = \begin{pmatrix} y_1 \\ \vdots \\ y_{t-1} \end{pmatrix}$$

is the column vector of relevance scores received so far, and

$$\mathbf{X}_t = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{t-1} \end{pmatrix}$$

is the matrix of row feature vectors of the images presented so far. Based on the estimated weight vector $\hat{\mathbf{w}}$, LinRel calculates an estimated relevance score $\hat{y}_I = \mathbf{x}_I \cdot \hat{\mathbf{w}}$ for each image I that has not already been presented to the user. To deal with the exploration-exploitation trade-off, LinRel selects for presentation not the image with largest estimated relevance score, but the image with the largest upper confidence bound for the relevance score. The upper confidence bound for an image I is calculated as $\hat{y}_I + c\hat{\sigma}_I$, where $\hat{\sigma}_I$ is an upper bound on the standard deviation of the relevance estimate $\hat{y}_I$. The constant c is used to adjust the confidence level of the upper confidence bound.

The rationale for using upper confidence bounds is that an image gets selected if (a) its relevance score is indeed large, or (b) the estimated relevance score is rather unreliable and the resulting confidence interval is large. Case (a) gives an exploitative choice, while case (b) improves the estimates and thus is explorative. It has been shown that upper confidence bounds are a versatile tool to balance exploration and exploitation in online selection problems [1, 4, 3]. In [3] rigorous bounds on the performance of LinRel are proven.

In each iteration t the regularised LinRel algorithm for $n = 1$ [5], calculates

$$\mathbf{a}_I = \mathbf{x}_I \cdot (\mathbf{X}_t^\top \mathbf{X}_t + \mu \mathbf{I})^{-1} \mathbf{X}_t^\top \tag{1}$$

for each image I and selects for presentation the image I_t which maximises

$$I_t = \arg\max_I \left\{ \mathbf{a}_I \cdot \mathbf{y}_t + \frac{c}{2} ||\mathbf{a}_I|| \right\} \tag{2}$$

for some specified constant $c > 0$.

3.1 Kernelization and selection of image collages

Kernel learning [21] can be integrated into the LinRel algorithm. A kernel learning algorithm learns a suitable metric between images in respect to a user query, by finding a good kernel function. Since only in each iteration the kernelized LinRel algorithm relies on a fixed kernel function, the integration of kernel learning into LinRel is very simple: LinRel calls the kernel learning algorithm at the beginning of each iteration and then uses the kernel matrix returned by the kernel learning algorithm. To kernelize LinRel [5],

$$\mathbf{a}_I = \big(\kappa(I, I_1) \ \cdots \ \kappa(I, I_{t-1})\big) \cdot (\mathbf{K}_t + \mu \mathbf{I})^{-1},$$

where $I_1, \ldots, I_{t-1}$ are the images selected in iterations $i = 1, \ldots, t-1$ and $\mathbf{K}_t$ is the Gram matrix

$$\mathbf{K}_t = (k(I_i, I_j))_{1 \le i,j \le t-1}.$$

Thus $\mathbf{a}_I$ can be calculated by using only the kernel function $\kappa(\cdot, \cdot)$. Since the selection rule (2) remains unchanged, this gives the kernelized version of LinRel.

Furthermore, three methods for the selection of image collages are used and evaluated in this paper:

1. Select the images $I_{t,1}, \ldots, I_{t,n}$ with maximal upper confidence bounds $\mathbf{a}_I \cdot \mathbf{y}_t + \frac{c}{2}||\mathbf{a}_I||$, where $\mathbf{a}_I = \mathbf{x}_I \cdot (\mathbf{X}_t^\top \mathbf{X}_t + \mu\mathbf{I})^{-1}\mathbf{X}_t^\top$ with $\mathbf{X}_t$ and $\mathbf{y}_t$,
2. Select image $I_{t,1}$ with the maximal upper confidence bound $\mathbf{a}_I \cdot \mathbf{y}_t + \frac{c}{2}||\mathbf{a}_I||$. Select the images $I_{t,2}, \ldots, I_{t,n}$ with maximal estimated relevance score $\mathbf{a}_I \cdot \mathbf{y}_t$,
3. For $k = 1, \ldots, n$ select image $I_{t,k}$ which maximises $\mathbf{a}_I \cdot \mathbf{y}_{t,k} + \frac{c}{2}||\mathbf{a}_I||$, where $\mathbf{a}_I = \mathbf{x}_I \cdot (\mathbf{X}_{t,k}^\top \mathbf{X}_{t,k} + \mu\mathbf{I})^{-1}\mathbf{X}_{t,k}^\top$,[7]

where the matrix $\mathbf{X}_{t,k}$ augments $\mathbf{X}_t$ by the feature vectors of the already selected images $I_{t,1}, \ldots, I_{t,k-1}$,

$$\mathbf{X}_t = \begin{pmatrix} \phi(\mathbf{x}_{I_{1,1}}) \\ \vdots \\ \phi(\mathbf{x}_{I_{t-1,n}}) \end{pmatrix}, \quad \mathbf{X}_{t,k} = \begin{pmatrix} \mathbf{X}_t \\ \phi(\mathbf{x}_{I_{t,1}}) \\ \vdots \\ \phi(\mathbf{x}_{I_{t,k-1}}) \end{pmatrix}.$$

The vector $\mathbf{y}_{t,k}$ augments the vector of previous outcomes $\mathbf{y}_t$ by the *expected outcomes* for the already selected images,

$$\mathbf{y}_t = \begin{pmatrix} y_1 \\ \vdots \\ y_{t-1} \end{pmatrix}, \quad \mathbf{y}_{t,k} = \begin{pmatrix} \mathbf{y}_t \\ \hat{y}_{t,1} \\ \vdots \\ \hat{y}_{t,k-1} \end{pmatrix},$$

with $\hat{y}_{t,j} = \hat{y}_{I_{t,j}}$.

The first method presents those n images with the highest upper confidence bounds. However, a drawback of relying only on the upper confidence values is that the similarities of the selected images are not considered. A simple method to avoid this problem is to use just one image for exploration (the second method) — by selecting the image with the maximum upper confidence bound $\hat{y}_I + c\hat{\sigma}_I$ — and selecting the remaining $n-1$ images exploitatively just according to the maximum estimated relevance scores $\hat{y}_I$. Thus this second method does as little exploration as possible, while the first method does the maximal possible exploration. The third method we consider covers the middle ground between these two extreme methods. It selects the images $I_1, \ldots, I_n$ for presentation sequentially, still relying on the upper confidence bounds, but when selecting image I_k it takes into account the exploration already done for images $I_1, \ldots, I_{k-1}$.

4 Multiple kernel learning to update the feature space

In multiple kernel learning (MKL) [15, 7, 2] we would like to solve a classification[8] problem using a combination of kernels to find (a) the weights of each kernel, and (b) the optimal dual weight vector of the combined kernel. The standard

[7] Combining this third method with kernel learning is a bit more involved, since for each selection $I_{t,k}$, $k = 1, \ldots, n$, the kernel learning algorithm needs to be called.

[8] Regression problems can also be solved, but we restrict ourselves to classification.

MKL framework has been proposed using the SVM and so we will also adopt this algorithm in our system. Given a set of kernels, the main goal of MKL is to find a non-uniform weighting of these kernels to help improve the classification task, over and above what could be achieved using a single kernel from the set.

More formally, assume we have a set of kernel functions $\mathcal{K} = \{\kappa_1, \ldots, \kappa_K\}$. Given a vector $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_K)$ of coefficients and kernel functions $\kappa_i(\cdot, \cdot)$ for $i = 1, \ldots, K$, we can define the following linear combination of kernel functions:

$$\kappa_{\boldsymbol{\eta}}(\cdot, \cdot) = \sum_{i=1}^{K} \eta_i \kappa_i(\cdot, \cdot).$$

In our case the kernels correspond to the $K = 11$ feature extraction methods from Table 1 and each component of $\boldsymbol{\eta}$ is the weighting of each of these features. From the previous section we know that the kernelized LinRel requires a kernel function between images. Therefore, after the user is presented with a set of images and classifies them as relevant or not, we will have access to these viewed images and their classifications $\in \{0, 1\}$. Using this information we can have the LinRelMKL algorithm:

$$\mathbf{a}_I(\boldsymbol{\eta}) = \big(\kappa_{\boldsymbol{\eta}}(I, I_1) \ \cdots \ \kappa_{\boldsymbol{\eta}}(I, I_{t-1})\big) \cdot (\mathbf{K}_t^{\boldsymbol{\eta}} + \mu \mathbf{I})^{-1},$$

where $I_1, \ldots, I_{t-1}$ are the images selected in iterations $i = 1, \ldots, t-1$ and $\mathbf{K}_t^{\boldsymbol{\eta}}$ is the Gram matrix

$$\mathbf{K}_t^{\boldsymbol{\eta}} = (\kappa_{\boldsymbol{\eta}}(I_i, I_j))_{1 \le i,j \le t-1}.$$

Thus $\mathbf{a}_I(\boldsymbol{\eta})$ can be calculated by using the MKL derived kernel function $\kappa_{\boldsymbol{\eta}}(\cdot, \cdot)$. It should be noted that the LinRelMKL algorithm can only learn after feedback has been given. In the CBIR protocol we follow, learning occurs after every page of images has been presented.

We follow an MKL approach recently proposed [13] that looks for a 1-norm 2-norm regularisation of the primal weight vectors in each feature space. The idea is to constrain the 2-norm of each weight vector $\mathbf{w}_k \in \mathbb{R}^m$ (note $\mathbf{w}_k$ is the weight vector in the kth feature space):

$$\min_{\mathbf{w}_k, \boldsymbol{\xi}} \ \underbrace{\lambda \left(\sum_{k=1}^{K} \|\mathbf{w}_k\|_2 \right)^2}_{1-\text{norm}} + \underbrace{(1-\lambda) \sum_{k=1}^{K} \|\mathbf{w}_k\|_2^2}_{2-\text{norm}} + C\|\boldsymbol{\xi}\|_1,$$

where $\lambda \in [0, 1]$ is a parameter, which gives us the standard 1-norm MKL regularisation when $\lambda = 1$ and the 2-norm when $\lambda = 0$, C is the SVM penalty parameter and $\boldsymbol{\xi}$ denotes the slack variables. Any values of λ between 1 and 0 will solve the MKL problem with a regularisation between a 1-norm and 2-norm. The justification of this form of regularisation is that in our CBIR system we expect that early on in the search, all features will need to be active, but as the search continues and more feedback is given we should be able to concentrate on a smaller number of relevant feature spaces (as $\lambda \to 1$).

We should point out that computing the kernels at each stage of a search page may become cumbersome for a large number of search pages. However, typically a CBIR search may only last 20 search pages, which, in our setting of presenting 15 images per search page (see Section 7) would correspond to kernel matrices of size 300×300. Therefore, the use of MKL together with LinRel in a CBIR system can be practical for the very reason that CBIR searches will never exceed a large number of search pages.

We have now described all of the principle components needed for our CBIR system when it has access to several different feature extraction methods. In the next section we discuss the integration of eye movement features.

5 Tensor learning for eye movements as new features

In this section we outline the approach taken when eye movement features are also available. We assume that the CBIR system has access to an eye tracking device, which tracks the eye movements of users whilst they view images. We follow the approach outlined in [11]. The idea is to create a tensor kernel between the image and eye movement features, and then to solve a tensor kernel SVM, in order to enrich our image feature space with eye movements. Note that we do not use eye movement features for relevance feedback but as an extra set of features. Our goal is to incorporate the information gained from the eye movements, into a new set of features and combine them with the content-based image features described in Table 1, using the MKL approach of the previous section.

The underlying idea for using tensors is for the creation of rich semantic representation that captures all possible relationships between features of two or more views (sources). This tensor representation creates an implicit correlation space in which we solve our learning problem – of course we must have an existing belief that there is some relationship between the different representations – which in our case is a valid assumption as we extract image and eye movement features of the *same* images.

An explicit mapping of the feature spaces and the tensor computation become computationally infeasible. However, recent papers have shown that the mapping can be made implicitly using the kernel trick [22, 17] – by simply taking the dot product between each individual kernel. For instance, let $\phi(\mathbf{x}_I)$ be the vector $\mathbf{x}_I$ mapped using feature mapping ϕ (*i.e.*, image features), and let $\psi(\mathbf{x}_I)$ be the vector $\mathbf{x}_I$ mapped using feature mapping ψ (*i.e.*, eye movement features). Therefore we have:

$$\begin{aligned}\langle \phi(\mathbf{x}_{I_i}) \circ \psi(\mathbf{x}_{I_i}), \phi(\mathbf{x}_{I_j}) \circ \psi(\mathbf{x}_{I_j}) \rangle &= \langle \phi(\mathbf{x}_{I_i}), \phi(\mathbf{x}_{I_j}) \rangle \langle \psi(\mathbf{x}_{I_i}), \psi(\mathbf{x}_{I_j}) \rangle \\ &= \kappa_\phi(I_i, I_j) \kappa_\psi(I_i, I_j),\end{aligned}$$

where $\circ$ denotes the tensor product. Hence, the Gram matrix $\mathbf{K}_\phi$ and $\mathbf{K}_\psi$ of each view is multiplied together using a component-wise product:

$$\mathbf{K}_{\phi \circ \psi} = \mathbf{K}_\phi \cdot \mathbf{K}_\psi,$$

where $\cdot$ denotes the component-wise product. Further to this the kernel matrix $\mathbf{K}_{\phi\circ\psi}$ is used to train a tensor kernel SVM [12] to generate a weight matrix W (see below). However, this weight matrix is composed of both views, and needs to be decomposed into one weight vector per view (*i.e.*, one for ϕ and one for ψ). This has been resolved by [12] who propose a novel singular value decomposition (SVD) like approach for decomposing the resulting tensor weight matrix into its two component parts, without needing to directly access the feature space. They demonstrate that the weight matrix can be decomposed into a sum of tensor products of corresponding weight components. Hence

$$W \approx W(D) = \sum_{d=1}^{D} \mathbf{w}_{\phi}^{d} {\mathbf{w}_{\psi}^{d}}^{\top},$$

where D is similar to the number of components in SVD and where each $\mathbf{w}_{\phi}^{d}, \mathbf{w}_{\psi}^{d}$ is a projection vector (like an eigenvector computed in PCA) in the dimension d. Furthermore, from the Representer theorem we know that each $\mathbf{w}_{\phi}^{d}, \mathbf{w}_{\psi}^{d}$ can be rewritten as a weighted combination of observed examples, such that:

$$\mathbf{w}_{\phi}^{d} = \sum_{i=1}^{m} \beta_i^d \phi(\mathbf{x}_{I_i}),$$
$$\mathbf{w}_{\psi}^{d} = \sum_{i=1}^{m} \gamma_i^d \psi(\mathbf{x}_{I_i}),$$

where $\boldsymbol{\beta}^d, \boldsymbol{\gamma}^d$ are dual variables (see [12] for decomposition details).

In our CBIR system we do not have the eye movement features for images not yet displayed to the user. Despite this restriction we are still interested in improving the semantic space by using both eye movements and image features during the phase when users have access to them, but then switching to using image features when deciding on which images to present next. Recall that we have access to K feature extraction methods (see Table 1) corresponding to feature maps $\phi_1, \ldots, \phi_K$, and we construct a kernel $\kappa_{\boldsymbol{\eta}}$ using a convex combination of the K feature spaces (see section 4). Following [11] we use the proposed decomposition and create a new feature representation for image features

$$\hat{\phi}(\mathbf{x}_{I_j}) = \left[\sum_{i=1}^{m} \kappa_{\boldsymbol{\eta}}(I_i, I_j) \beta_i^d \right]_{d=1}^{D},$$

where we now have a kernel $\hat{\kappa}_{\boldsymbol{\eta}}(I, I) = \langle \hat{\phi}(\mathbf{x}_I), \hat{\phi}(\mathbf{x}_I) \rangle$ constructed from these new image features $\hat{\phi}(\mathbf{x})$. Hence, the LinRelMKLTensor algorithm is:

$$\hat{\mathbf{a}}_I(\boldsymbol{\eta}) = \left(\hat{\kappa}_{\boldsymbol{\eta}}(I, I_1) \ \cdots \ \hat{\kappa}_{\boldsymbol{\eta}}(I, I_{t-1}) \right) \cdot (\hat{\mathbf{K}}_t^{\boldsymbol{\eta}} + \mu \mathbf{I})^{-1},$$

where $I_1, \ldots, I_{t-1}$ are the images selected in iterations $i = 1, \ldots, t-1$ and $\hat{\mathbf{K}}_t^{\boldsymbol{\eta}}$ is the Gram matrix

$$\hat{\mathbf{K}}_t^{\boldsymbol{\eta}} = (\hat{\kappa}_{\boldsymbol{\eta}}(I_i, I_j))_{1 \leq i,j \leq t-1}.$$

6 The final CBIR system

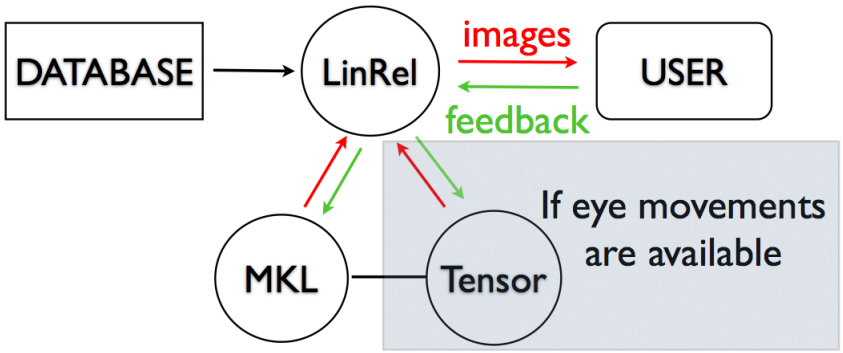

Fig. 3: Our CBIR system: the LinRelMKL algorithm and LinRelMKLTensor when eye movements are available. The red lines indicate features (inputs) and the green lines indicate relevance-feedback given by the user.

Figure 3 describes pictorially the components of our proposed CBIR system. We have the "DATABASE" which contains all images we have access to. The LinRel algorithm has direct access to the images from the database. Furthermore, after presenting some images to the "USER", the system receives feedback on the images that are relevant/non relevant by the user. This feedback is then passed to "MKL" and "Tensor" (when eye movements are available) in order to learn an enriched feature space utilising image and eye movement features as outlined in Sections 4 and 5 (*i.e.*, LinRelMKL and LinRelMKLTensor). This protocol is repeated until the user is satisfied with the search, and retrieved the image(s) they were looking for. In general, any number of images may be presented to the user but in our experiments we will present 15 images per search page.

When "MKL" is not utilised, but eye movements are available for "Tensor" then we will assume that LinRel receives a sum of uniform weighted kernels *i.e.*, $\boldsymbol{\eta} = (1/K, \ldots, 1/K)$. We call this algorithm LinRelTensor.

7 Experiments

The database used for images was the PASCAL VOC 2007 challenge database [10] which contains over 9000 images, each of which contains at least 1 object from a possible list of 20 objects such as cars, trains, cows, people, *etc.*,. There were $T = 23$ users in the experiments, and participants were asked to view

twenty pages, each of which contained fifteen images. Their eye movements were recorded by a Tobii X120 eye tracker [23] connected to a PC using a 19-inch monitor (resolution of 1280x1024). The eye tracker has approximately 0.5 degrees of accuracy with a sample rate of 120 Hz and used infrared lens to detect pupil centres and corneal reflection. Figure 4 shows an example image presented to a user along with the eye movements (in red).

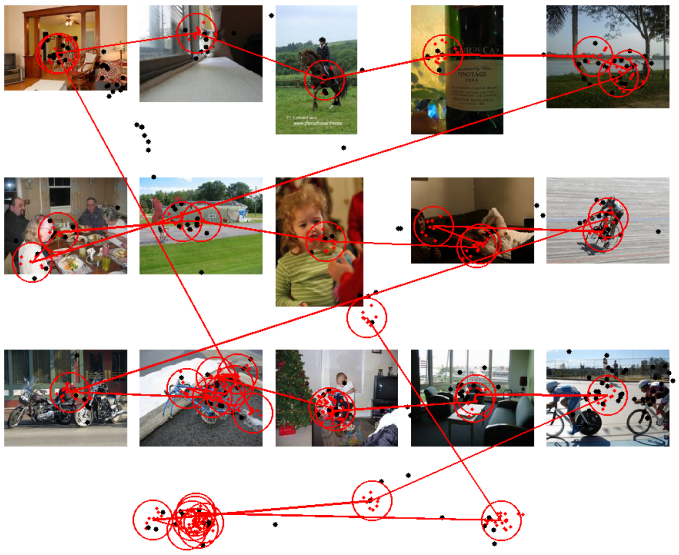

Fig. 4: An example page shown to a participant. Eye movements of the participant are shown in red. The red circles mark fixations and small red dots correspond to raw measurements that belong to those fixations. The black dots mark raw measurements that were not included in any of the fixations.

Each participant was asked to carry out one search task among a possible of three, which included looking for a Bicycle (8 users), a Horse (7 users) or some form of Transport (8 users). Any images within a search page that a user thought contained images from there search, they marked as relevant using mouse clicks.

The eye movement features extracted from the Tobii eye tracker can be found in Table 1 of [16] and was collected by [20]. Most of these are motivated by features considered in the literature for text retrieval studies – however, in addition, they also consider more image-based eye movement features. These are computed for each full image and based on the eye trajectory and locations of the images

in the page. They cover the three main types of information typically considered in reading studies: fixations, regressions, and re-fixations. The number of features extracted were 33, computed from: (i) raw measurements from the eye tracker; (ii) fixations estimated from the raw measurements using the standard ClearView fixation filter provided with the Tobii eye tracking software, with settings of: "radius 50 pixels, minimum duration 100 ms".

For each subjects feedback and eye movement data, we train our system to find the most relevant images for the given search task (*i.e.*, Train, Horse or Transport). We randomly permute the images and choose 15 images initially to present to our CBIR system. Our system then trains the MKL (if used) and tensor kernel SVM (if used) to pass a new kernel to LinRel, which then chooses the next set of 15 images to be presented. This is repeated until all images are presented. We measure our success using Average Precision – in other words, by looking at the number of retrieved images with respect to their rank. For our results we used several random seeds to initialise the first set of images presented to the CBIR system, and averaged over them.

For our experiments we use linear kernels[9] constructed using the features presented in Table 1 and the eye movement features described above. Table 2 presents the Average Precision results, where we used a leave-one-subject-out strategy to optimise the parameters for the LinRel algorithm, for various values[10] of c and μ. Furthermore, we selected method 2 from Subsection 3.1 which seemed to generate the best performance out of the three methods outlined there. Parameters for all other algorithms were kept fixed. The LinRel column is the kernelized LinRel algorithm using a sum of uniform weighted linear kernels. The LinRelMKL column corresponds to a kernel being learnt for LinRel using MKL.[11] The LinRelTensor column corresponds to no MKL being learnt (hence, a default uniform weighting of kernels) but tensor kernel SVM[12] being trained to find a new feature space. Finally, the LinRelMKLTensor column corresponds to both the MKL and tensor kernel SVM being used to find an appropriate kernel for LinRel.

We can see by comparing the LinRel and LinRelMKL methods that applying MKL can improve the Average Precision. However, when eye movements are available then LinRelTensor and LinRelMKLTensor are clearly superior and help improve the search results, with LinRelMKLTensor being better than LinRelMKL with a statistical significance (Wilcoxon signed test) of $p = 0.05$.

[9] Nonlinear kernels can also be used, but we chose linear kernels for their simplicity and for proof of concept.

[10] $\mu = (10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3})$, $c = (10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3})$.

[11] MKL used fixed parameter $\lambda = 0.5$ throughout the experiments, and a penalty parameter $C = 1$.

[12] penalty parameter $C = 1$.

Table 2. Average Precision results for 23 subjects (users) searching for Bicycle (user 1-8), Horse (user 9-15) and Transport (16-23).

Subject	LinRel	LinRelMKL	LinRelTensor	LinRelMKLTensor
1	0.320045	0.335052	0.498231	**0.505099**
2	0.377128	0.379353	**0.519415**	0.516407
3	0.240924	0.248338	0.409046	**0.415642**
4	0.358075	0.368579	**0.501031**	0.498201
5	0.375263	0.388169	**0.521616**	0.515576
6	0.368568	0.374660	**0.523426**	0.507518
7	0.373300	0.369395	**0.514916**	0.498389
8	0.358244	0.375362	0.514891	**0.512478**
9	0.369601	0.365979	0.614555	**0.624961**
10	0.322203	0.322497	**0.522267**	0.510010
11	0.349775	0.343399	**0.567960**	0.566198
12	0.345682	0.344114	**0.557776**	0.555769
13	0.348333	0.357742	**0.585215**	0.574832
14	0.326431	0.319506	**0.497876**	0.481860
15	0.327132	0.334904	0.525686	**0.530794**
16	0.393561	0.389895	0.616144	**0.624988**
17	0.330865	0.333592	0.578743	**0.592307**
18	0.424255	0.429772	0.649531	**0.658993**
19	0.372035	0.372421	0.602936	**0.621422**
20	0.307158	0.309784	0.560447	**0.580845**
21	0.383495	0.385278	0.611653	**0.627579**
22	0.321837	0.329154	0.531695	**0.563895**
23	0.313631	0.309275	0.568597	**0.577140**
Average	0.348154	0.351575	0.547550	**0.550474**

8 Conclusions

We described a novel content-based image retrieval system (CBIR) using relevance-feedback, which views the problem as an exploration-exploitation problem – by using a kernelized version of the LinRel algorithm together with multiple kernel learning (when several different feature extraction methods are available) we demonstrated that we can combine these two algorithms together to learn different feature weightings (LinRelMKL). We also proposed a novel technique of combining the kernel constructed using MKL (on image features) and the kernel constructed using eye movements. This kernel was then trained using the tensor kernel SVM to yield a new feature space combining eye movement and image feature spaces. We called this algorithm LinRelMKLTensor and observed that the search results improved over all other methods (on average). Hence, when eye movement features are available then they should be used in order to improve CBIR systems.

In the current work we only used linear kernels constructed from the image and eye movement features. In future work we plan to use a parameterised

family of kernels such as Gaussian kernels. Initial results on Gaussian kernels for the kernelized LinRel algorithm have reported good results. Furthermore, by constructing several different Gaussian kernels per feature extraction method our model would find a weighted combination of important kernels using the MKL component of the system. This would also give us a much larger set of kernels – shown to be effective at improving MKL [6].

Also, we used the discrete values of the feedback received from users to train the MKL and tensor kernel SVM. However, we could of course apply a regression version of MKL and a tensor kernel support vector regression (for instance), by using the real valued relevance estimates provided by users. This would also more closely resemble the optimisation problem solved by LinRel, which is itself a linear regression algorithm.

Another line of further research is to use the eye movements for (implicit) relevance feedback. The system we have outlined in this paper *only* uses explicit feedback from mouse clicks to determine the relevance of images. However, by utilising the eye movements we could predict the relevance of images based on what users look at. This would make the CBIR system much more automated and require less explicit interaction by the user. We have carried out preliminary work using our system with such a mechanism in place, and the results are encouraging. We plan to discuss these results in future studies.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under *grant agreement* no. 216529, Personal Information Navigator Adapting Through Viewing, PinView. This work was supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

References

1. R. Agrawal. Sample mean based index policies with O(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, 27(4):1054–1078, 1995.
2. A. Argyriou, C. A. Micchelli, and M. Pontil. Learning convex combinations of continuously parameterized basic kernels. In *Computational Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 338–352. Springer, 2005.
3. P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.
4. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
5. P. Auer, A. Leung, Z. Hussain, and J. Shawe-Taylor. Report on using side information for exploration-exploitation trade-offs. PinView FP7-216529 Project Deliverable Report D4.2.1, December 2009.

6. F. R. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 105–112. 2009.
7. F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6, New York, NY, USA, 2004. ACM.
8. Y. Chen, X. S. Zhou, and T. Huang. One-class SVM for learning in image retrieval. *Proceedings of International Conference on Image Processing 2001*, 1:34–37, 2001.
9. R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40:5:1–5:60, 2008.
10. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
11. D. R. Hardoon and K. Pasupa. Image ranking with implicit feedback from eye movements. In *Proceedings of ETRA 2010: ACM Symposium on Eye-Tracking Research & Applications*, pages 291–298. ACM, 2010.
12. D. R. Hardoon and J. Shawe-Taylor. Decomposing the tensor kernel support vector machine for neuroscience data with structure labels. *Machine Learning Journal: Special Issue on Learning From Multiple Sources*, 79(1-2):29–46, 2010.
13. Z. Hussain, K. Pasupa, C. J. Saunders, and J. Shawe-Taylor. Basic metric learning. PinView FP7-216529 Project Deliverable Report D3.1, December 2008.
14. J. Laaksonen and V. Viitaniemi. Evaluation of pointer click relevance feedback in picsom. PinView FP7-216529 Project Deliverable Report D1.2, August 2008.
15. G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.
16. K. Pasupa, C. Saunders, S. Szedmak, A. Klami, S. Kaski, and S. Gunn. Learning to rank images from eye movements. In *HCI '09: Proceeding of the IEEE 12th International Conference on Computer Vision (ICCV'09) Workshops on Human-Computer Interaction*, pages 2009–2016, 2009.
17. S. Pulmannová. Tensor products of hilbert space effect algebras. *Reports on Mathematical Physics*, 53(2):301–316, 2004.
18. J. Rocchio. *Relevance Feedback in Information Retrieval*, pages 313–323. 1971.
19. Y. Rui and T. Huang. Optimizing learning in image retrieval. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 236–243, 2000.
20. C. Saunders and A. Klami. Database of eye-movement recordings. Technical Report Project Deliverable Report D8.3, PinView FP7-216529, 2008. available on-line at http://www.pinview.eu/deliverables.php.
21. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, U.K., 2004.
22. S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez. Learning via linear operators: Maximum margin regression; multiclass and multiview learning at one-class complexity. Technical report, University of Southampton, 2005.
23. L. Tobii Technology. Tobii Studio Help. http://studiohelp.tobii.com/StudioHelp_1.2/.
24. S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM.