Neurocomputing 248 (2017) 80–93

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Utilising Kronecker Decomposition and Tensor-based Multi-view Learning to predict where people are looking in images

Kitsuchart Pasupa[a,*], Sandor Szedmak[b]

[a] Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand
[b] Department of Computer Science, Aalto University, Aalto FI-00076, Finland

ARTICLE INFO

Article history:
Received 27 June 2016
Revised 9 November 2016
Accepted 15 November 2016
Available online 8 March 2017

Keywords:
Multi-view learning
Missing data
Tensor algebra
Maximum margin learning
Eye movements
Kronecker decomposition

ABSTRACT

Eye movement data collection is very expensive and laborious. Moreover, there are usually missing values. Assuming that we are collecting eye movement data from a set of images viewed by different users, there is a possibility that we will not able to collect the data of every user from every image–one or more views may not be represented in the image. We assume that the relationships among the views can be learnt from the whole collection of views (or items). The task is then to reproduce the missing part of the incomplete items from the relationships derived from the complete items and the known part of these items. Using certain properties of tensor algebra, we showed that this problem can be formulated consistently as a regression type learning task. Furthermore, there is a maximum margin based optimisation framework in which this problem can be solved in a tractable way. This problem is similar to learning to predict where a person is looking in an image. Therefore, we proposed an algorithm called "Tensor-based Multi-View Learning"(TMVL) in this paper. Furthermore, we also proposed a technique for improving prediction by introducing a new feature set obtained from Kronecker decomposition of the image fused with user's eye movement data. Using this new feature can improve prediction performance markedly. The proposed approach was proven to be more effective than two well-known saliency detection techniques.

1. Introduction

Many researchers have paid attention to image understanding that allows computer to capture the meanings of images in the same way as humans do. One of the challenging tasks in image understanding is saliency prediction. Visual saliency is a property of locations or object in the visual world, e.g., in images. If an object is salient, it stands out from its neighbours, with a high probability of being able to draw humans' attention to it.

It is very important to learn which parts humans tend to look at in scenes or images. Saliency prediction is useful in many applications–such as graphic design, web design, and human computer interaction–because it enables designers to evaluate their visual design quality. Many methods of saliency modelling have been proposed [1,2]. More methods can be found in a recent survey paper that covered 256 publications related to saliency object detection [3]. Saliency models can be divided into two categories: supervised and unsupervised learning based models.

Itti et al. [1] and Harel et al. [2] investigated bottom-up visual saliency (i.e., low level image feature that does not involve supervised information) [1,2]; unfortunately, human gazes do not usually match the map [4] because they are highly influenced by any image related tasks. If users are requested to view images without having been given a particular task, their gazes will be automatically directed by low-level image feature. In the case that users have been given a clear and specific task, their eye movements will be controlled by the content of the image. Consequently, top-down visual features are the features that should be considered [5]. Another approach is supervised learning based saliency model. It utilises eye movement data which are mapped with image features [6–9].

In order to learn where humans tend to look at in images, an eye tracker is required to collect eye movement data. In real-world scenarios, eye movement data collection is tedious, laborious, and expensive. Moreover, data loss is inevitable as (i) an eye tracker may temporarily lose track of a subject because he or she is moving during the experiment, and (ii) a subject may fail to respond to all of the tasks. Consequently, we aimed to estimate missing eye movement data from the available data on the same task. It is similar to learning to predict where users tend to look at based on

* Corresponding author.
E-mail addresses: kitsuchart@it.kmitl.ac.th, k.pasupa@gmail.com (K. Pasupa).

http://dx.doi.org/10.1016/j.neucom.2016.11.074
0925-2312/

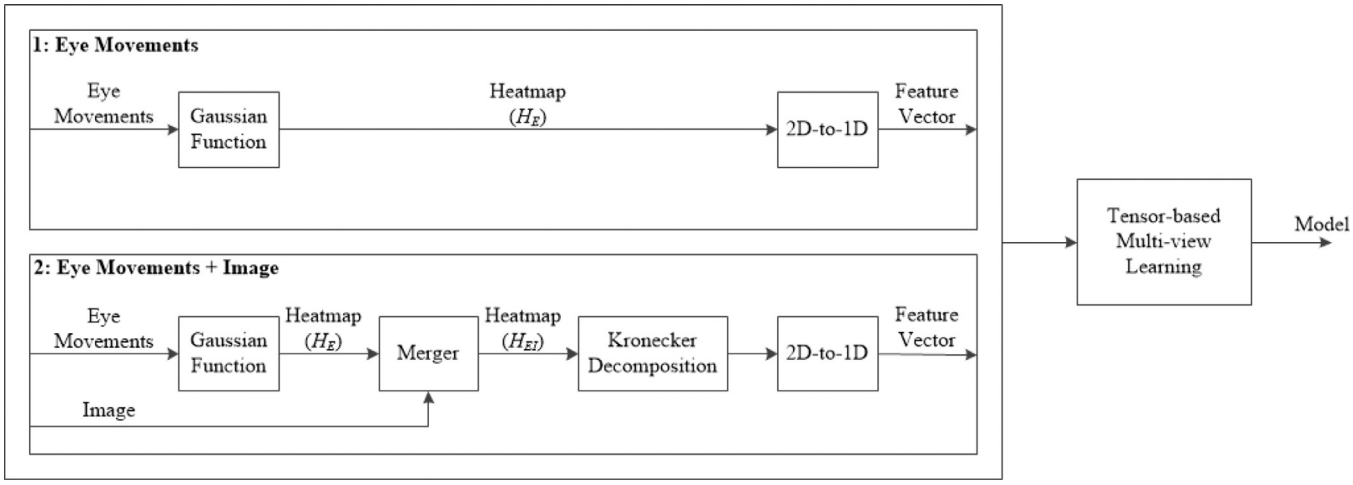


Fig. 1. Main components and data flow in this work. There are two sets of features which are eye movement information only and image fused with eye movement information.

their previous eye movement data on other images and on other available users' eye movement data on the considered image. This leads to the learning scenario introduced in this paper. It is built with a general assumption that multiple views of a problem are available. It is not always possible to observe all of the views in a realistic scenario; therefore, this problem can be cast as a multi-view learning problem with missing data. In this scenario, we assume that initially there is a subset of training samples in which a complete set of views can be observed, but later on probably only some random subsets of views can be collected.

The goal of the learning task is to estimate the values of missing views in each sample. This scenario can occur in a real experiment. This type of problem is a generalisation of classical supervised learning problems such as regression. Face recognition is one of the applications that can be considered under this framework (when some parts or views of the faces are unknown because of occlusion, for example). In developing the learning framework, we made two mild assumptions: (i) there is a reasonable large number of observations (samples) where all of the views are known, therefore, a learning procedure can be realised; and (ii) from incomplete observations, at least one view is available. We made no assumption about the distribution of the missing views, but any prior knowledge about the missing data can be exploited to improve the estimation of their values.

In this paper, we introduced a formulation that can be considered as a generalised regression problem in which missing values are estimated from available views and their relationships are extracted from training samples. Assuming that the missing views of a sample are output $\mathbf{y}$ and the known parts are input $\mathbf{x}$, then we have $\mathbf{y} \Leftarrow \mathbf{W}\mathbf{x}$, where $\mathbf{W}$ is a linear operator that learns from complete data and describes the relationships between different views. The difficulty of this kind of regression arises from the fact that the output and input may vary among the sample items. We proposed a "Tensor-based Multi-view Learning (TMVL)"algorithm to handle the problem of incomplete view. Providing a tractable learning algorithm, TMVL is based on properties of tensor algebra and maximum margin-based optimisation framework. Tensor decomposition has already been used in some missing data problems, e.g., [10,11], but their settings are different from ours. Liu et al. (2013) investigated a low-rank tensor technique based on tensor-trace norm minimisation problem in image reconstruction [10], while Chen and Grauman (2014) proposed a probabilistic tensor model for inferring human appearance from unseen viewpoints [11]. In this paper, we show that our proposed method can estimate missing eye movements, which can be exploited for predicting where humans are likely to look at in images.

We also propose a novel approach for fusing eye movement information with image features in order to enhance prediction performance. There are many pieces of evidence suggesting that fusing low-level image features with eye movement improves prediction accuracy [12–15]. Here, we employed factors derived from Kronecker decomposition of image fused with eye movement data to represent each view in TMVL.

The outline of the paper is as follows: Section 2 describes all methods proposed in this work, including TMVL algorithm, and tensor decomposition. Section 3, shows the performance results of our proposed methods on a real-world dataset. Finally, the conclusion of our study is presented in Section 4.

2. Methodology

This section explains the methods proposed in this work: Section 2.1 describes the TMVL algorithm, its algebraic framework, and the corresponding optimisation problem. Section 2.2 explains the procedure for decomposing images as tensors, followed by an approach for combining images with eye movement data in Section 2.3. A model-training process framework, including a data processing pipeline for two sets of features, is visually summarised in Fig. 1.

2.1. Tensor-based Multi-view Learning Algorithm

As previously mentioned, we had a set of complete views of our samples that we used as training set, and a test set in which the views were not complete (missing randomly). We aimed to fill in the missing views for each sample. An example of multi-view learning problem with missing data is shown in Fig. 2.

2.1.1. Algebraic framework

Let us denote $\mathcal{R} = \{1, \ldots, n_R\}$ as the set of indices of the views considered. In our model, each of these views has a corresponding linear vector space $\mathcal{Z}_r,\ r \in \mathcal{R}$ over real numbers. The dimensions of these spaces are denoted by $\mathrm{Dim}(\mathcal{Z}_r) = d_r,\ r \in \mathcal{R}$. The set $\mathcal{J}_R = \{j_1, \ldots, j_{n_R}\}$ comprises the indices of the samples within each of the spaces corresponding to the views, enumerating the components of the vectors chosen from the space corresponding to the views. The range of these indices is equal to the number of dimensions of the corresponding space.

	Views			
Source spaces:	$\mathcal{Z}_1$	$\mathcal{Z}_2$	$\mathcal{Z}_3$	$\mathcal{Z}_4$
	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
Complete items:	$\mathbf{z}_1^1$	$\mathbf{z}_1^2$	$\mathbf{z}_1^3$	$\mathbf{z}_1^4$
(Training)	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathbf{z}_m^1$	$\mathbf{z}_m^2$	$\mathbf{z}_m^3$	$\mathbf{z}_m^4$
Incomplete items:	$\mathbf{z}_{m+1}^1$	$\mathbf{z}_{m+1}^2$	·	$\mathbf{z}_{m+1}^4$
(Test)	·	·	$\mathbf{z}_{m+2}^3$	$\mathbf{z}_{m+2}^4$
	·	$\mathbf{z}_{m+3}^2$	·	·
	$\mathbf{z}_{m+4}^1$	·	·	$\mathbf{z}_{m+4}^4$
	·	$\mathbf{z}_{m+5}^2$	$\mathbf{z}_{m+5}^3$	·
	·	$\mathbf{z}_{m+6}^2$	$\mathbf{z}_{m+6}^3$	$\mathbf{z}_{m+6}^4$
	$\mathbf{z}_{m+7}^1$	·	$\mathbf{z}_{m+7}^3$	·
	·	·	·	$\mathbf{z}_{m+8}^4$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Fig. 2. Graphical representation of the multi-view learning framework for a four-view learning problem.

A sample is chosen out of direct products of these spaces, and each sample item consists of as many vectors as the number of views,

	Views:			
Linear vector spaces:	$\mathcal{Z}_1$	$\dots$	$\mathcal{Z}_{n_R}$	
	$\Downarrow$	$\dots$	$\Downarrow$	
Sample:	$\mathbf{z}_i^1$	$\dots$	$\mathbf{z}_i^{n_R}$	$i = 1, \dots, m.$

The product space of the views is given by the tensor product of the spaces, $\mathcal{Z} = \bigotimes_{r\in\mathcal{R}} \mathcal{Z}_r$. This construction forms the algebraic framework of our solution, see [16,17] and the references therein for more details.

If we are given two tensor products of vectors then the following contraction operator [., .] can be defined over them as

$$[\bigotimes_{q\in\mathcal{Q}} \mathbf{u}^q, \bigotimes_{r\in\mathcal{R}} \mathbf{v}^r] = \prod_{q\in\mathcal{Q}\cap\mathcal{R}} \langle \mathbf{u}^q, \mathbf{v}^q \rangle \bigotimes_{q\in\mathcal{Q}\setminus\mathcal{R}} \mathbf{u}^q \bigotimes_{r\in\mathcal{R}\setminus\mathcal{Q}} \mathbf{v}^r, \tag{1}$$

where the inner product is computed for all common indices. When the two index sets are coincident then the following well known identity can be used to unfold the inner products of the tensor products as

$$\langle \bigotimes_{q\in\mathcal{Q}} \mathbf{z}_i^r, \bigotimes_{q\in\mathcal{Q}} \mathbf{z}_j^q \rangle = \prod_{q\in\mathcal{Q}} \langle \mathbf{z}_i^q, \mathbf{z}_j^q \rangle. \tag{2}$$

This identity states that the inner product of tensor products of vectors is equal to the product of the inner product of these vectors.

This interpretation of the indices is compatible with the notations commonly used in tensor algebra, namely, with the so-called "Einstein summation convention". The symbol of summation Σ is omitted and summation has to be carried out over all indices which are denoted by the same symbol. Since we use this strategy to denote views and algorithmic iterations which are not tensor indices, we choose to handle summations with special care by making them explicit via the contraction operator [., .]. Furthermore, we assume an orthogonal representation of the indices, hence in turn, there is no need to make distinction between covariant and contravariant indices.

In the learning problem, we use a linear operator, a tensor, that is an element of the dual space of $\mathcal{Z}$, the space of linear functionals defined on $\mathcal{Z}$, namely

$$\mathbf{W} \in \mathcal{Z}^*, \ \mathbf{W} = [W_{\mathcal{J}_R}] = [W_{j_1,\dots,j_{n_R}}],$$

where $\mathcal{Z}^*$ denotes the dual space of all possible linear functionals defined on $\mathcal{Z}$.

We can write up Frobenius type inner products between the linear operator $\mathbf{W}$ and the tensor product of vectors of the views as

$$\langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F = \sum_{j_1,\dots,j_{n_R}} W_{j_1,\dots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathbf{z}_{ij_r}^r. \tag{3}$$

In similar fashion, we can compute the Frobenius norm of $\mathbf{W}$ by

$$\|\mathbf{W}\|_F = \Big(\sum_{j_1,\dots,j_{n_R}} W_{j_1,\dots,j_{n_R}}^2 \Big)^{\frac{1}{2}}. \tag{4}$$

In the next step, the set of views is partitioned into two arbitrary parts,

$$\mathcal{R}_X \subset \mathcal{R}, \ \mathcal{R}_Y = \mathcal{R} \setminus \mathcal{R}_X,$$

we term the views occurring in $\mathcal{R}_X$ as inputs, and the views in $\mathcal{R}_Y$ can be handled as outputs. Corresponding to either $\mathcal{R}_X$ or $\mathcal{R}_Y$, the set of indices belonging to each view has to be split apart,

$$\mathcal{J}_X \subset \mathcal{J}_R, \ \mathcal{J}_X = \{j_r, \ r \in \mathcal{R}_X\}, \ \mathcal{J}_Y = \mathcal{J}_R \setminus \mathcal{J}_X.$$

Fixing a partition, a contraction of $\mathbf{W}$ can be defined as

$$W_{J_Y} = W_{J_R \setminus J_X} = \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \stackrel{\text{def}}{=} \sum_{j_r\in\mathcal{J}_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r, \tag{5}$$

where the components of $\mathbf{W}$ are summed only over the input views.

Consequently, the relationship between the inputs and the outputs can be described by the following inner product

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F \stackrel{\text{def}}{=} \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathbf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r. \tag{6}$$

It provides a similarity measure between the outputs and the projection of the inputs by the linear operator $\mathbf{W}$. If the norm of $\mathbf{W}$ is fixed, then this inner product takes a greater value if the angle between the direction of the outputs and the projection of the inputs is smaller, thus the correlation between them is greater. If both the inputs and the outputs are normalised to the same length then this similarity measure implies small distance as well.

Based on these definitions, we can derive a simple but fundamental Lemma:

Lemma 1. *For all partitions $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the inner products*

$$\langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F$$

have the same value, namely

$$\langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F.$$

Proof. We need to unfold only the corresponding definitions of the inner products that give the next chain of equalities

$$\begin{aligned} \langle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r \rangle_F &= \sum_{J_Y} \prod_{s\in\mathcal{R}_Y} \mathbf{z}_{ij_s}^s \sum_{J_X} W_{J_R} \prod_{r\in\mathcal{R}_X} \mathbf{z}_{ij_r}^r \\ &= \sum_{j_1,\dots,j_{n_R}} W_{j_1,\dots,j_{n_R}} \prod_{r\in\mathcal{R}} \mathbf{z}_{ij_r}^r \\ &= \langle \mathbf{W}, \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F. \end{aligned}$$

□

This Lemma shows that the value of the inner product of the tensor products is invariant on the partitioning of the views into inputs and outputs.

2.1.2. The optimisation problem

To force a high similarity between the projected inputs and the outputs taken out of a fixed partition of the views, a "Support Vector Machine"-style, maximum-margin-based optimisation problem is formulated for the regression task. Please note the earlier application of the framework [18,19]:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} \quad & \mathbf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \langle \underbrace{\bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s}_{\text{Outputs}}, \mathbf{W} \underbrace{\bigotimes_{r\in\mathcal{R}_X} \mathbf{z}_i^r}_{\text{Inputs}} \rangle_F \geq 1-\xi_i, \\ & \xi_i \geq 0,\ i = 1,\ldots,m, \end{aligned} \tag{7}$$

where $C > 0$ is penalty constant.

The form is similar to the Support Vector Machine case with two notable exceptions: (i) the outputs are no longer binary labels, $\{-1,+1\}$, but vectors of an arbitrary linear vector space, and (ii) the normal vector of the separating hyperplane is reinterpreted as a linear operator projecting the inputs into the space of the outputs.

The regularisation term in the objective function forces the projections of the inputs and the outputs to be similar with respect to their inner products. When the inputs and the outputs are normalised, they live on a sphere in both corresponding spaces, hence we solve the problem by using the structure of Spherical rather than Euclidean geometry.

Based on Lemma 1, we state the next theorem:

Theorem 2. *For all partitions, $\mathcal{R}_X, \mathcal{R}_Y$ of $\mathcal{R}$ the optimisation problem (7) is equivalent to the following one:*

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{W}\|_F^2 + C\sum_{i=1}^m \xi_i \\ \text{w.r.t.} \quad & \mathbf{W} \text{ tensor} \in \mathcal{Z}^*,\ \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \langle \mathbf{W}, \textstyle\bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r \rangle_F \geq 1-\xi_i,\ i = 1,\ldots,m, \\ & \xi_i \geq 0,\ i = 1,\ldots,m. \end{aligned} \tag{8}$$

This equivalence holds true if the inputs and the outputs are partitioned independently for every sample item.

Proof. We can reformulate the constraints by following Lemma 1 that proves the statement. □

This fact guarantees that the linear operator $\mathbf{W}$ has a universal property that *it is independent of the way how the views are grouped into inputs and outputs*, thus it consistently characterises the underlying multi-view learning problem.

The seemingly complex problem represented by (8) can be solved via a simple Lagrangian dual:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\boldsymbol{\alpha}'\left(\mathbf{K}_1 \bullet \cdots \bullet \mathbf{K}_{n_R}\right)\boldsymbol{\alpha} - \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^m \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{9}$$

where

$$(\mathbf{K_r})_{ij} = \langle \mathbf{z_i^r}, \mathbf{z_j^r} \rangle,\ r \in \mathcal{R},\ i,j \in \{1,\ldots,m\} \tag{10}$$

are kernels corresponding to each of the views. In the formulation of the Lagrangian dual, we exploited the identity given by (2).

The $\bullet$ operator expresses the element-wise, Hadamard, product of matrices. This dual can be solved in a straightforward way for very large scale applications[1]. After the dual variables were computed, the optimum solution for the universal linear operator becomes

$$\mathbf{W} = \textstyle\sum_{i=1}^m \alpha_i \bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r. \tag{11}$$

[1] The website of the authors provides an open source implementation to this problem.

In the test phase, known and unknown views are considered as inputs and outputs, respectively. The output can be estimated in the following way:

$$\begin{aligned} \Big(\bigotimes_{s\in\mathcal{R}_y} \mathbf{z}^s\Big) \sim \mathbf{W} \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}^r &= \sum_{i=1}^m \alpha_i [\bigotimes_{r\in\mathcal{R}} \mathbf{z}_i^r, \bigotimes_{r\in\mathcal{R}_X} \mathbf{z}^r] \\ &= \sum_{i=1}^m \alpha_i \prod_{r\in\mathcal{R}_X} \langle \mathbf{z}_i^r, \mathbf{z}^r \rangle \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s \\ &= \sum_{i=1}^m \beta_i \bigotimes_{s\in\mathcal{R}_Y} \mathbf{z}_i^s, \end{aligned} \tag{12}$$

where

$$\beta_i = \alpha_i \textstyle\prod_{r\in\mathcal{R}_X} \langle \mathbf{z}_i^r, \mathbf{z}^r \rangle,\ i = 1\ldots,m. \tag{13}$$

Thus, the prediction is a linear combination of the corresponding known outputs.

2.1.3. Computational complexity of the proposed method

For estimation of the computational complexity of the problem presented in (9), one can recognise that (9) is equivalent to the dual problem of an unbiased Support Vector Machine (SVM). Consequently, the problem in (9) can be solved by applying the same type of methods that should have the same complexity depending on the sparsity of the included kernels, see a discussion about this in [20]. The basic task of the SVM is to separate two classes of output data by a hyperplane. The settings of the SVM contain a sample $\{y_i, \mathbf{x}_i\},\ y_i \in \{-1,+1\},\ \mathbf{x}_i \in \mathcal{X}(=\mathbb{R}^n)$ and the separating hyperplane which is defined by its normal vector $\mathbf{w}$. Furthermore, input vectors might be embedded into a feature space, $\mathcal{H}_X$, via a function $\phi : \mathcal{X} \to \mathcal{H}_X$, where it is assumed that $\mathcal{H}_X$ is a Hilbert space.

The corresponding primal optimisation problem of the SVM is formulated as

$$\begin{aligned} \min \quad & \tfrac{1}{2}\|\mathbf{w}\|_2^2 + C\mathbf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} \quad & \mathbf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m, \\ \text{s.t.} \quad & y_i \mathbf{w}'\phi x_i \geq \mathbf{1} - \xi_i, \\ & \boldsymbol{\xi} \geq \mathbf{0}, i = 1,\ldots,m, \end{aligned} \tag{14}$$

while the dual problem of the SVM has the form (see for example in [21]),

$$\begin{aligned} \min \quad & \tfrac{1}{2}\boldsymbol{\alpha}'(\mathbf{y}\mathbf{y}' \bullet \mathbf{K}_X)\boldsymbol{\alpha} - \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} \in \mathbb{R}^m, \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}. \end{aligned} \tag{15}$$

After introducing the notation $\mathbf{K} = (\mathbf{y}\mathbf{y}' \bullet \mathbf{K}_X)$ in the SVM dual, and similarly $\mathbf{K} = \left(\mathbf{K}_1 \bullet \cdots \bullet \mathbf{K}_{n_R}\right)$ in the dual of the multi-view learning problem (9), we arrive at the same optimisation problem, thus the computational complexity of the proposed learning problem is the same as the complexity of the SVM.

2.1.4. Non-linear relations

Another consequence of the equivalence of the dual optimisation problems of the proposed method and the SVM is that the kernel trick can be applied here as well (see [21] for more details on background of kernel trick). Since both the dual problem (9) and the prediction (12) contain all input feature vectors as components of the corresponding inner products knowing only a positive semi-definite kernel matrix suffices for carrying out computations in the training and in the prediction as well. More concretely, let all input features – the representation of the image parts in this application – be implicitly embedded into a feature space by these functions,

$$\boldsymbol{\phi}_r : \mathcal{Z}_r \to \mathcal{H}_r,\ r \in \mathcal{R}_X, \tag{16}$$

where $\mathcal{H}_r$ is a Hilbert space for each r. Then we can write up the elements of the kernels as,

$$(\mathbf{K_r})_{ij} = \langle \boldsymbol{\phi}_r(\mathbf{z}_i^r, \boldsymbol{\phi}_r(\mathbf{z}_j^r)\rangle, r \in \mathcal{R}_X,\ \ i, j = \{1, \ldots, m\}. \tag{17}$$

For example, Gaussian or Polynomial kernels can be chosen on top of the available linear features.

2.2. Decomposing images as tensors

Images expressed by matrices can be represented as a product of the factors computed by Kronecker decomposition. This Kronecker decomposition, after a reordering of the elements of the image matrix, can be carried out by singular value decomposition (SVD). That kind of transformation of images can reveal the structure of the images, e.g., edges, and corners, and can yield a high level compression of the image matrices as well. In case of colour images, tensors can replace the matrices in order to capture the third dimension of the colours.

2.2.1. Kronecker decomposition of matrices

Let us consider a real 2-dimensional (2D) image decomposition of which we can expect that the points nearby within continuous

$$\mathbf{X} = \mathbf{A} \otimes \mathbf{B},$$

$$\left[\begin{array}{cc|cc|cc} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} \\ \hline x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} \\ \hline x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} \\ x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} \end{array}\right] = \left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{array}\right] \otimes \left[\begin{array}{cc} b_{11} & b_{12} \\ b_{21} & b_{22} \end{array}\right],$$

can be reordered into

$$\widetilde{\mathbf{X}} = \widetilde{\mathbf{A}} \otimes \widetilde{\mathbf{B}},$$

$$\left[\begin{array}{ccccccccc} x_{11} & x_{13} & x_{15} & x_{31} & x_{33} & x_{35} & x_{51} & x_{53} & x_{55} \\ x_{12} & x_{14} & x_{16} & x_{32} & x_{34} & x_{36} & x_{52} & x_{54} & x_{56} \\ x_{21} & x_{23} & x_{25} & x_{41} & x_{43} & x_{45} & x_{61} & x_{63} & x_{65} \\ x_{22} & x_{24} & x_{26} & x_{42} & x_{44} & x_{46} & x_{62} & x_{64} & x_{66} \end{array}\right] = \left[\begin{array}{c} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{array}\right] \otimes \left[\begin{array}{ccccccccc} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{array}\right],$$

2D blocks can relate stronger to each other than points in the 1-dimensional rows or columns. A question is, "can the SVD decomposition provide 2D blocks instead of vectors?", achieve this end, a Kronecker decomposition is applied.

The Kronecker product of a matrix $\mathbf{X}$ can be expressed as

$$\mathbf{X} = \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{1,1}\mathbf{B} & A_{1,2}\mathbf{B} & \cdots & A_{1,n_A}\mathbf{B} \\ A_{2,1}\mathbf{B} & A_{2,2}\mathbf{B} & \cdots & A_{2,n_A}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m_A,1}\mathbf{B} & A_{m_A,2}\mathbf{B} & \cdots & A_{m_A,n_A}\mathbf{B} \end{bmatrix} \tag{18}$$

where $\mathbf{A} \in \mathbb{R}^{m_A \times n_A}$, $\mathbf{B} \in \mathbb{R}^{m_B \times n_B}$, $m_X = m_A \times m_B$, and $n_X = n_A \times n_B$.

In the Kronecker decomposition, the second component, $\mathbf{B}$, might be interpreted as a 2D filter of the image represented by the matrix $\mathbf{X}$. We can try to find a sequence of those filters by applying the procedure presented in Algorithm 1.

The question is: given $\mathbf{X}$, how can we compute the optimum solution $\mathbf{A}$ and $\mathbf{B}$ for the problem,

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X}^{(k)} - \mathbf{A} \otimes \mathbf{B}\|^2_{Frobenius}? \tag{19}$$

Algorithm 1 Calculate the Kronecker decomposition of a matrix.

Require: matrix $\mathbf{X}$, number of iteration n,
Require: size of $\mathbf{A} \in \mathbb{R}^{m_A \times n_A}$, size of $\mathbf{B} \in \mathbb{R}^{m_B \times n_B}$
Ensure: $(\mathbf{A}^{(1)}, \mathbf{B}^{(1)}), \ldots, (\mathbf{A}^{(n)}, \mathbf{B}^{(n)})$
1: $\mathbf{X}^{(1)} = \mathbf{X}$
2: **for** $k = 1$ to n **do**
3: $\quad \mathbf{A}^{(k)}, \mathbf{B}^{(k)} = \arg\min_{\mathbf{A},\mathbf{B}} \|\mathbf{X}^{(k)} - \mathbf{A} \otimes \mathbf{B}\|^2_{Frobenius}$
4: $\quad \mathbf{X}^{(k)} = \mathbf{X}^{(k)} - \mathbf{A}^{(k)} \otimes \mathbf{B}^{(k)}$, ## deflation
5: **end for**

2.2.2. Kronecker decomposition as SVD

We can make an important observation that the algorithm solving the tensor decomposition problem does not depend directly on the order of the elements of the matrix, thus a permutation of the indexes, i.e. reordering of the columns and/or the rows, preserves the same solution. Based on this observation, the Kronecker decomposition can be computed via the SVD (see [22] for more details). An example that illuminates how the reordering of the matrix $\mathbf{X}$ can solve the Kronecker decomposition problem is demonstrated here:

The matrices in the Kronecker product

where the blocks of $\mathbf{X}$ and the matrices $\mathbf{A}$ and $\mathbf{B}$ are vectorised in a row-wise order.

We observe that $\tilde{\mathbf{X}} = \tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$ can be interpreted as the first step in the SVD algorithm where we might apply the substitutions $\sqrt{s}\mathbf{u} = \tilde{\mathbf{A}}$ and $\sqrt{s}\mathbf{v} = \tilde{\mathbf{B}}$. The proof that this reordering provides a correct solution to the Kronecker decomposition can be found in [22].

The main steps of the Kronecker decomposition can be summarised as follows:

1. Reorder (reshape) the matrix,
2. Compute the SVD decomposition,
3. Compute the approximation of $\tilde{\mathbf{X}}$ by $\tilde{\mathbf{A}} \otimes \tilde{\mathbf{B}}$
4. Invert the reordering.

This kind of Kronecker decomposition is often referred as the Nearest Orthogonal Kronecker Product as well [22].

We can extend this procedure to higher order tensors represented by higher order arrays (see the review paper of [23]). Then, this method can be applied on the following objects:

- colour images, where three matrices express the RGB layers, a tensor of order 3, e.g. $1024 \times 1024 \times 3$,

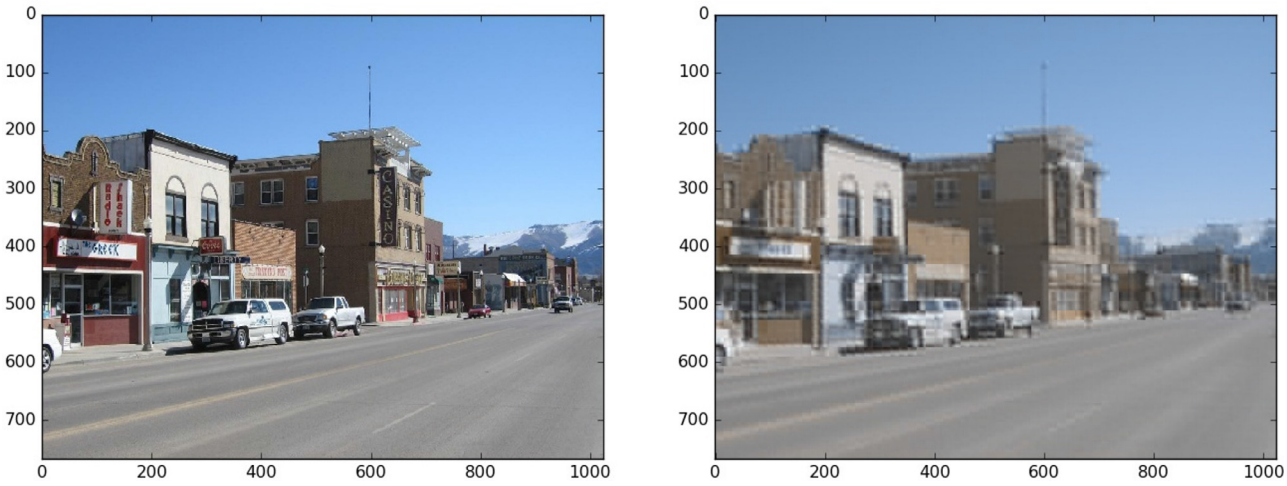


Fig. 3. The right image is a recovered image from the Kronecker decomposition of the original image on the left. The factors were (48, 64, 1), (16, 16, 3) with 6 singular values. The compression ratio was 102.4.

- video stream of grey-scale images, where the third dimension is time,
- video stream of colour images, where the third dimension is colour, and the fourth dimension is time.

The Kronecker decomposition presented above can be extended further to include more than two factors (more details, alternative approaches and applications can be found in [23]). The Kronecker decomposition algorithm, as shown in Algorithm 1, implements a non-linear polynomial approximation of the target matrix or a higher order tensor. The degree of the applied polynomial is equal to the number of included factors.

2.2.3. A set theoretic approach to reordering

A matrix representation as an ordered collection of elements can be reinterpreted by using the language of set theory. Let $\mathbf{X}$ be a matrix with size $m \times n$. For the sake of simplicity, we assume that the elements $[X_{ij}]$ of $\mathbf{X}$ are real numbers. The structure of the matrix $\mathbf{X}$ can be described as a set $\mathcal{X}$ of the elements $\{X_{ij}\}$ with cardinality mn on which two equivalence relations, $\mathscr{R}_r$ and $\mathscr{R}_c$, are imposed, one based on the rows and the other based on the columns, i.e. two elements are equivalent with respect to $\mathscr{R}_r$ if they are in the same row, and equivalent with respect to $\mathscr{R}_c$ if they are in the same column.

Now, the reordering of the matrix elements $\mathcal{X}$ is expressible by imposing another kind of equivalence relations that classify the elements into different classes. These equivalence relations have to satisfy the following rules: all of the equivalence classes within the relation $\mathscr{R}$ have to have the same cardinality. The equal cardinality rule implies that the product of the number of the equivalence classes $N_{\mathscr{R}}$ and the common cardinality of the classes, $\text{card}_{\mathscr{R}}$, is equal to the cardinality of $\mathcal{X}$, namely to mn. Then, the reordering of the matrix elements can be carried out by sorting the equivalence classes into columns and rows of a new matrix.

Note that the equivalence relation $\mathcal{R}$ can be decomposed further into a series of equivalence relations $(\mathcal{R}_1, \ldots, \mathcal{R}_N)$. This decomposition can be realised in a recursive way.

1. Let the set of classes of $\mathcal{R}_1$ be given by $\mathcal{C}(\mathscr{R})_1, \ldots, \mathcal{C}(\mathscr{R})_{N_{R_1}}$, with common cardinality card_{R_1}.
2. For every class of relations, $\mathscr{R}_k$ apply the same type of equivalence relations, $\mathscr{R}_{k+1}$. Since the cardinality of the classes of $\mathscr{R}_k$ is the same, this relation can be applied uniformly. Consequently, the cardinality of the classes and the number of the classes in $\mathscr{R}_{k+1}$ are the same for all classes of $\mathscr{R}_k$.

Table 1
Examples of possible patterns of decomposition of a grey-scale image with 1024 × 1024.

Component Size	Singular values	Full size	Compression ratio
(32, 32), (32, 32)	10	$s = 10 * 2 * 32^2$	$\frac{1024^2}{s} = 51.2$
(16, 16), (16, 16), (4, 4)	20	$s = 20 * (2 * 16^2 + 4^2)$	$\frac{1024^2}{s} = 99.29$

By reversing the recursive classification of the elements, we can build a tensor $\mathbf{T}$ of order $N + 1$ by starting on the classes of the $\mathscr{R}_N$.

Since the internal structure of the matrix $\mathbf{X}$ is not directly exploited, only the set of its elements are classified. The reordering procedure described above can be applied on any tensor of arbitrary order. Thus, any tensor can be reordered into another tensor of arbitrary order. An example of order one tensor, vectorisation, can be given by an equivalence relation where all elements fall into the same equivalent class.

This interpretation of the reordering leads us to the realm of *Algebraic Statistics*, and within that realm, to the *Combinatorial Design Theory* [24,25]. The Combinatorial Design Theory addresses the problem of how to build systems of finite sets that satisfy certain requirements of symmetries as stated in the classical theory of *Latin squares*. These theories play a very important role in creating balanced statistical experimental designs, especially for medical tests.

2.2.4. Compression

Furthermore, the tensor decomposition can provide a very high level compression of images. Some examples are presented here. The compression ratio is computed by dividing the number of elements of the image matrix with the total number of elements of the components in the decomposition.

The original image matrix is given by integers in the range of $0, \ldots, 255$ for both grey-scale and RGB colour images. The decomposition happens in the space of real numbers, but after having been decomposed, the real numbers in the components can be rescaled and transformed into the original integer interval.

Let the size of a grey-scale image be equal to (1024, 1024), then we can have various patterns of decompositions as shown in Table 1.

The components provide a tightly-compressed signature for the image as illustrated in Fig. 3.

(a) Image without compression (b) Image compressed by Kronecker decomposition (c) Image compressed by SVD

Fig. 4. An example of image compressed by the proposed and a conventional technique when the compression ratio was 100.

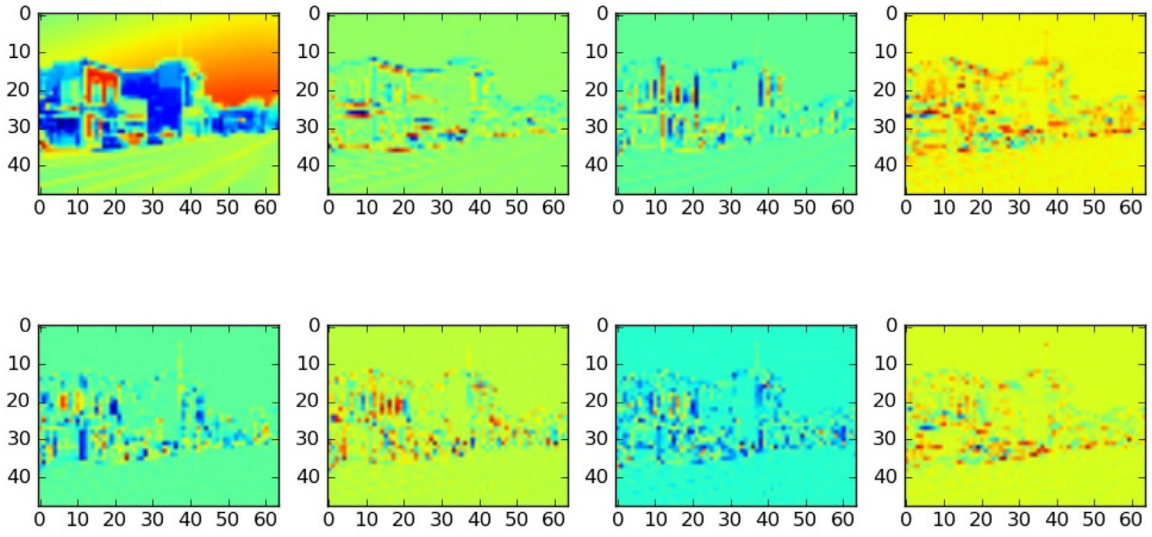


Fig. 5. First level of the Kronecker decomposition in case of the first 8 largest singular values.

The Kronecker-decomposition-based features can be compared with those from a conventional compression technique with SVD and the well-known SIFT features [26]. Let the size of an RGB colour image be (640, 960, 3) and the sizes of the components in the tensor decomposition be (10, 15, 1), (8, 8, 1), (8, 8, 3), and let 45 singular values be computed, then $45 * (10 * 15 + 8^2 + 3 * 8^2)$ 8-bit data items can be obtained with a high compression ratio of $\sim$ 100. This is equivalent to (i) using image compression with SVD with 4 singular values which gives $((640 * 4) + 4 + (940 * 4)) * 3$ data items and (ii) $\sim$ 36 SIFT feature vectors with 128 real valued components, where we assume that these real numbers can be represented by 32 bits. The tensor decomposition can recover a good approximation of the original image and the colour information while the conventional technique cannot do so, as shown in Fig. 4. Although the 36 SIFT points may be able to capture some particular characteristic points, they cannot recover the main structure of the entire image at all.

2.2.5. Interpretation of image components

The components provided by the tensor decomposition can be endowed with a practical interpretation. Let us discuss the two-component case where the image matrix **X** is expressed as a Kronecker product of two other matrices **A** and **B**. Since the second factor **B** in the product is forced to be the same for all positions, it has to be equal to a nonlinearly aggregated matrix. This factor is shifted around within the image and it is only scaled by elements of the matrix **A**. Thus, **B** can be interpreted as an image filter.

In Fig. 5, the first level components belonging to the sequence of decreasing singular values are of the compressed image shown in Fig. 3. This image is factorised into two levels and the corresponding low level components are presented in Fig. 6. Some characteristic patterns can be recognised from these factors. The factor belonging to the second singular vector represents a horizontal edge filter and the factor belonging to the third singular value yields a vertical line filter. In this way, the image decomposition finds the boundaries of the critical regions, edges, and corners in which most of the structural information concentrates, and as mentioned earlier, it also produces a very highly compressed skeleton of the data.

Since in every step the decomposition processes the residue of the previous step, it predicts those parts that have hardly been approximated earlier, thus in every step a new layer of the structure is discovered. In images, these layers are first the flat areas, then edges of different directions–vertical, horizontal, and slant–corners of different kind, and the higher order singularities of the intensity surface. This kind of incremental approach resembles a boost in which hardly predictable sample items receive larger weights.

2.2.6. Relation to known saliency detection approaches

The generally applied saliency detectors are built on estimated derivatives of the intensity function, I, of an image (see for example [27]). These derivatives can provide information about the locations that the intensity changes significantly faster than a given threshold. The general first order edge and corner detection algorithms consist of two phases: the first phase uses a Gaussian filter

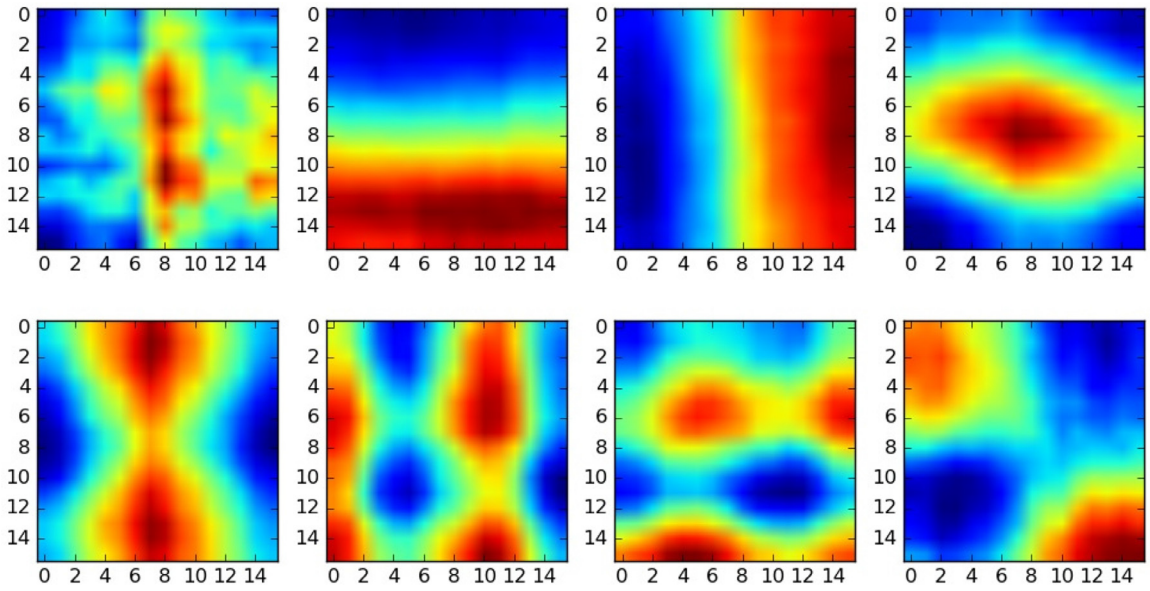


Fig. 6. Second level of the Kronecker decomposition in case of the first 8 largest singular values.

to smooth out the noise caused by non-differentiable representation of the image, and the second phase estimates the gradients of the image intensity. The higher order methods also exploit the second derivative, the Hessian, of the intensity image. Estimation of the derivatives can be performed by applying image filters on the image intensity, e.g. Laplacian, Sobel and Prewitt.

For example *Harris-Laplace detector* relies on the first order partial derivatives, I_x, I_y, of the intensity measured in the local neighbourhood of a given image point

$$\mathbf{A}(x) = \sum_{i,j} w_{ij} \begin{bmatrix} I_x^2(\mathbf{x}) & I_x I_y(\mathbf{x}) \\ I_x I_y(\mathbf{x}) & I_y^2(\mathbf{x}) \end{bmatrix},$$

$$R = \det(\mathbf{A}) - \alpha \ \mathrm{trace}^2(\mathbf{A}).$$

Similarly the *Hessian affine region detector* exploits the second partial derivatives of the Gaussian-filtered intensity, the Laplacian of the Gaussian (*LoG*) representation,

$$\mathbf{H}(x) = \begin{bmatrix} L_{xx}(\mathbf{x}) & L_{xy}(\mathbf{x}) \\ L_{xy}(\mathbf{x}) & L_{yy}(\mathbf{x}) \end{bmatrix},$$

$$L(x) = g(\cdot|0, \sigma_I) \otimes I(x),$$

$$LoG = \mathrm{trace}(\mathbf{H}),$$

where L is the Gaussian-filtered intensity, and g is a Gaussian filter with a parameter σ_I.

The Kronecker decomposition based algorithm iteratively reproduces the partial derivatives of the intensity function via higher order polynomial approximation of successive error terms. During this iterative procedure, the algorithm automatically selects, in the least-square sense, the best fitting filters as the last factor of the Kronecker product. In this way, it gives an approximation of the *multivariate Taylor series* of the intensity image that provides an estimation of the full series of those partial derivatives that are exploited in well-known saliency detectors. It is also important to mention that the global optimality of the Kronecker decomposition allows us to select only regions where the derivatives change the most and eliminate flat, redundant ones. This selective property leads to high level compression of the information needed to describe the variation of the intensity. For example, Harris detector based methods focus only on local features without forcing globally optimal extraction.

2.3. Combining images with eye movements

The images processed in our eye movement experiments were assumed to be RGB colour images represented by 3 parallel matrices and indexed by row and column coordinates of the pixels in the images. Since the sizes of the images varied, all of them were transformed into the same common size in the learning procedure. That common size was 50×50.

The eye movements in all of the experiments were given as a list of coordinate pairs consisting of the image related row and column indices. When the eyes moved beyond the image, those eye movement coordinates were represented by invalid, e.g., negative values. Therefore, the coordinate pairs of the eye movement had to be filtered to extract only those points that corresponded to valid image pixels. When the sizes of the images were transformed to the same size, that transformation applied to the eye movement coordinates as well.

When eye movement coordinates were combined with an image, we needed to deal with the uncertainty of eye positions caused by measurement errors and movements of the eyes and the head that were probably not related to the image. That uncertainty was handled by applying Gaussian smoothing, a spatial filter, to each point of eye movement. The Gaussian filter was centred at a point $\mathbf{u}$ with a width parameter σ assigns to each image point $\mathbf{x}$,

$$g(\mathbf{x}|\mathbf{u}, \sigma) = e^{-\frac{\|\mathbf{x}-\mathbf{u}\|^2}{2\sigma}}. \tag{20}$$

The centres of the Gaussian filters were localised at the observed points of the eye movements. Since all image points could be connected by the filter to all eye movement points, as many filter values were assigned to each image point as the number of observed eye movement points. To aggregate the filter values, we applied the following function on all of the image points $\mathbf{x}$:

$$g_A(\mathbf{x}) = \max_{\mathbf{u}} g(\mathbf{x}|\mathbf{u}, \sigma). \tag{21}$$

The function g_A assigned to each point of the image, $\mathbf{x}$, a Gaussian filter value that belonged to the closest point of the eye movement. That Gaussian filter can well represent foveation in human vision [28].

In Fig. 7, the eye movement trajectories on a stimuli image of six contributing users are presented. We were able to have some impression on the similarity and the variation of the responds of

(a) User 1 (b) User 2 (c) User 3

(d) User 4 (e) User 5 (f) User 6

Fig. 7. The eye movements of six different users on a sample image.

Fig. 8. Eye movement data overlaid on the image (left) and images merged with eye movement data (right) of two users.

the users looking at the same image and trying to perform the same task.

Fig. 8 shows images that were already merged with eye movements data. These images would subsequently be decomposed by a Kronecker product. The decomposition of the Kronecker product on the image and the combined images with eye movement data are shown in Fig. 9.

3. Performance evaluations and discussions

We evaluated our TMVL algorithm on a publicly available eye tracking dataset [4]. The dataset contains eye tracking data of 15 different users on 1003 images. Each image consists of three-second free-view trajectories of different users. In order to encourage users to pay attention to the task, they were memory-tested at the end of the data collection of 100 images.

In our experiment, only eight users were randomly selected; hence, there were eight views in this setting. Each view was represented by a heatmap of each user. Heatmap quantifies the degree of importance of parts of image; a higher probability of an important part is implied by a higher density of eye movements on that part. We investigated two sets of heatmap input features.

1. Heatmap generated from eye movement data alone (H_E): A user's heatmap was created by convolving a Gaussian kernel with each eye fixation point.
2. Heatmap generated from eye movement data and the image (H_{EI}): This set of features was generated by a Kronecker decomposition of an image combined with eye movement data as described in Section 2.3 with σ of 2. We extracted features with two components of Kronecker product from a grey scale processed image. The sizes of the high (first) ($\mathbf{A}$) and low (second) ($\mathbf{B}$) level component were (10,10), and (5,5), respectively with 20 singular values. The feature vector became $\left[\begin{array}{cccc} \mathbf{A}_1 \otimes \mathbf{B}_1 & \mathbf{A}_2 \otimes \mathbf{B}_2 & \ldots & \mathbf{A}_{20} \otimes \mathbf{B}_{20} \end{array} \right]$.

All heatmaps were normalised to unit norm. To compare prediction performances, we used three performance matrices as follows:

1. Area under the receiver operating characteristic (AUROC) is one of commonly used performance metrics for comparing heatmaps and eye fixations[2]. It is based on an ROC curve that can be computed by varying the threshold of the predicted heatmap. A pixel is predicted as a target when its heatmap value is greater than the threshold, while it is classified as a background when the value is below the threshold. AUROC ranges from 0 (complete mismatch) to 1 (perfect match).

[2] Available for download at http://www.vision.caltech.edu/~harel/share/gbvs.php.

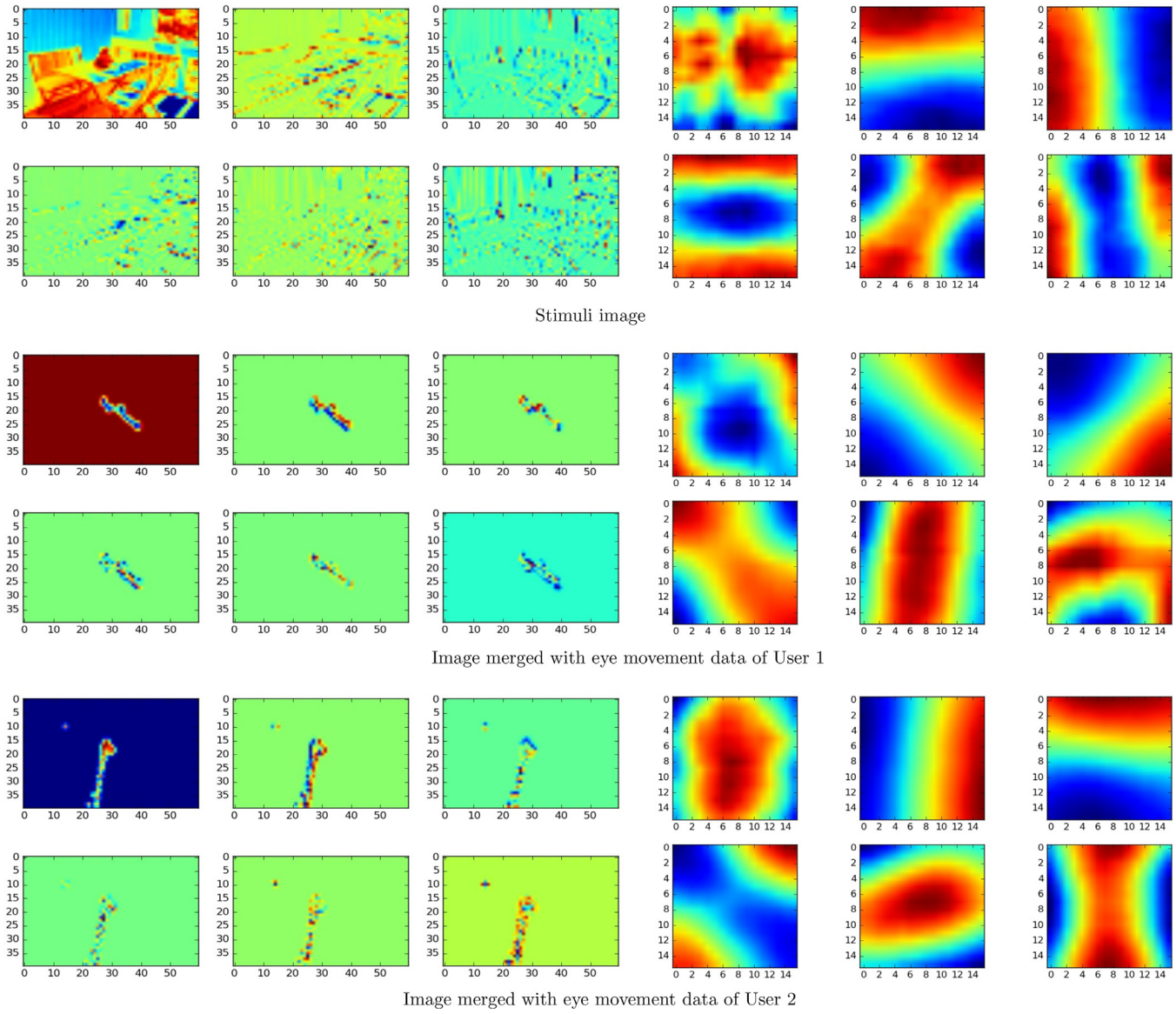


Fig. 9. Kronecker decomposition of the original stimuli image and the corresponding eye movement data of two users; two Kronecker factors and six singular values were computed.

2. Correlation indicates the degree of linear relationship between two maps. It ranges from −1 (perfect correlation but in the opposite direction) to +1 (perfect correlation). Zero indicates no correlation between the two maps.
3. Jensen–Shannon divergence (JSD) was used to identify the dissimilarity between two distributions. It is based on Kullback–Leibler divergence (KLD) which can capture a certain kind of non-linear, entropy type and dependency. KLD is defined as,

$$D_{KL}(P||Q) = \sum_{i=1} \ln\left(\frac{P_i}{Q_i}\right) P_i \tag{22}$$

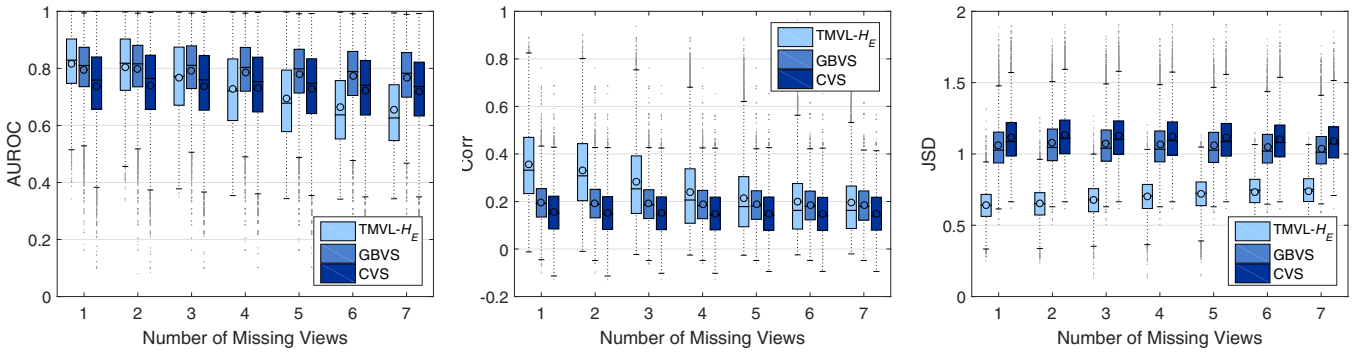
where P, Q are the probability distributions. JSD is symmetric while KLD is not [29]. Square root of JSD has matrix properties. The more similar the two objects are, the smaller the value of JSD is, and vice versa. JSD is defined as,

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \tag{23}$$

where $M = \frac{1}{2}(P+Q)$.

Here, linear kernel function was used. Model selection was performed with five-fold cross validation based on AUROC. We examined two scenarios of the test sets: (i) {1-7} missing views were randomly selected and (ii) one fixed view was missing for each run. The experiments were run 10 times with different random data splits.

Although our scenario is of a supervised learning approach, but its setting is different from those mentioned earlier in the introduction. All previous works used a saliency map of average locations fixated by all users as a global model. In our scenario, we instead used individual fixation maps and focused on user models. Our assumption was that there were some correlations between users' eye movement behaviours. We aim to predict the individual maps. To the best of our knowledge, there was no available technique designed for this kind of scenario, so we compared our proposed method with baseline methods in a similar way that we did in one of our previous works [30]. The baseline methods were two well-known saliency map models: Conventional Visual Saliency (CVS) [1], and Graph-Based Visual Saliency (GBVS) [2].



(a) AUROC (b) Correlation (c) JSD

Fig. 10. A boxplot comparison of all methods when randomly selected {1–7} missing views were considered.

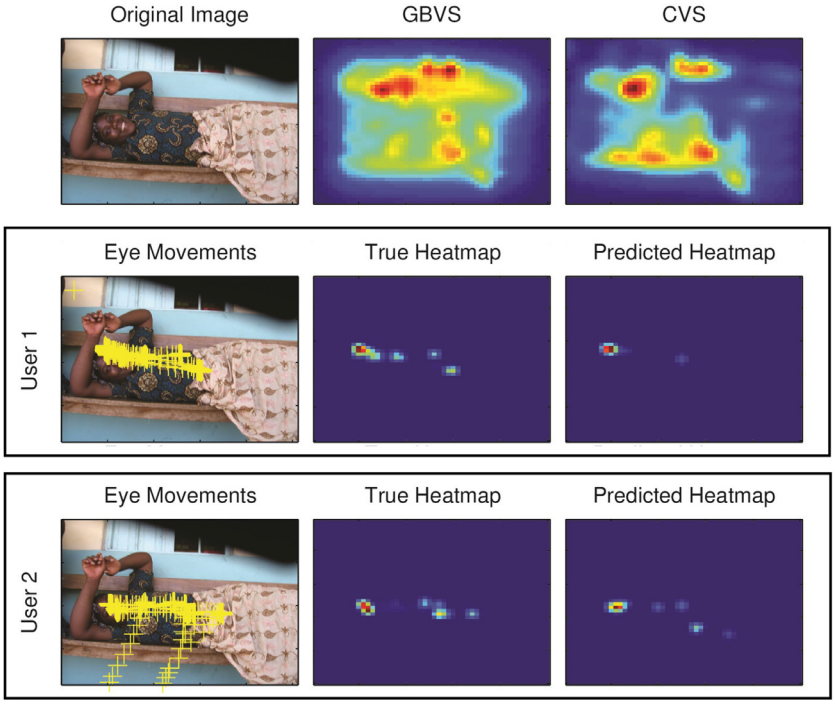


Fig. 11. True and TMVL-predicted heatmaps of an eight-view problem with two views missing (User 1 and 2) compared to those predicted by baseline methods–GBVS and CVS.

3.1. Randomly select missing views

When the number of missing views increased, AUROC and correlation decreased for all algorithms as shown in Fig. 10a and 10b, respectively. On the other hand, JSD increased when there was an increase in the number of missing views as shown in Fig. 10c. When the number of missing views increased, TMVL-H_E performance was dramatically reduced while GBVS and CVS's performances were slightly decrease. TMVL-H_E only used eye movement data, therefore, the prediction performances highly depended on number of available view, while GBVS and CVS used image information. TMVL-H_E's performances on AUROC were better than those on GBVS when there were 1–2 missing views and better than those on CVS when there were 1–3 missing views. Unfortunately, TMVL-H_E performances on AUROC were worse than those on GBVS and CVS in other cases as shown in Fig. 10a. However, TMVL-H_E was still able to achieve better average correlation and JSD than GBVS and CVS were in all cases as shown in Fig. 10b and 10c, respectively.

Table 2
Performance matrices of all of the methods on the image in Fig. 11. Bold values indicate the best performance achieved in each user.

User	Method	AUROC	Corr	JSD
1	TMVL	**0.8065**	**0.7160**	**0.3726**
	GBVS	0.6543	0.0783	1.2755
	CVS	0.7358	0.0792	1.2454
2	TMVL	**0.7812**	**0.6900**	**0.4064**
	GBVS	0.6707	0.0985	1.1863
	CVS	0.6983	0.0981	1.1627

Fig. 11 shows an example of prediction by GBVS, CVS, and our proposed algorithm when two views were missing. It can be seen that GBVS and CVS failed to predict where users were looking at. Both algorithms put attention on the woman's arm and the windows while users actually focused on her face. Clearly, TMVL was more effective than GBVS and CVS for all three performance matrices as shown in Table 2.

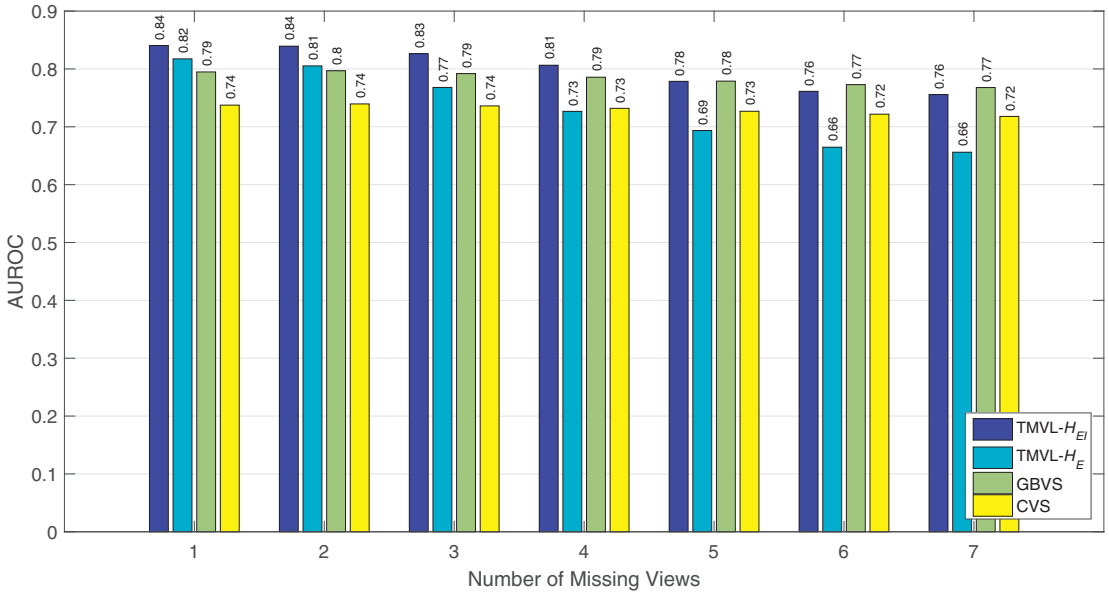


Fig. 12. A comparison between the proposed method and two existing saliency prediction methods in the case of randomly selected {1–7} missing views.

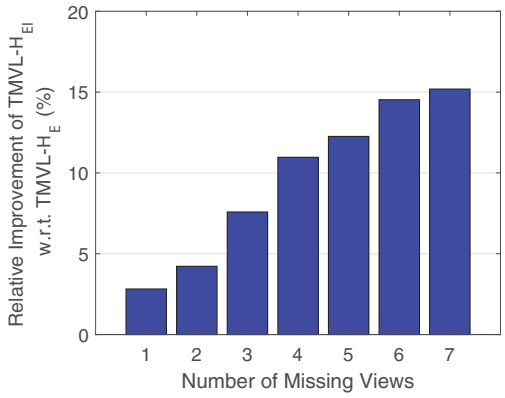


Fig. 13. Relative improvement of AUROC using H_{EI} compared to using H_E in TMVL.

We improved prediction performance by using H_{EI} as feature vectors and compared the prediction results with those from other methods, as shown in Fig. 12. TMVL-H_{EI} outperformed CVS in all cases but was only better than GBVS in {1–4}-missing-view cases. TMVL was comparable to GBVS when five views were missing but was worse than GBVS in the case of {6–7} missing views. It should be noted that we only evaluated the performances on AUROC as it compared the predicted heatmaps directly with eye movement data for all of the methods. Correlation and JSD compare between predicted heatmaps to target heatmaps but target heatmaps for TMVL with H_E and with H_{EI} were different. Hence, we were not able to directly compare these matrices. It is clear that using H_{EI} that combined images with eye movement information could dramatically enrich the performance on AUROC over using H_E as shown in Fig. 13. AUROC was improved at 2.82% in 1-missing-view case and up to 15.20% in seven-missing-view case.

Fig. 14 shows an example of predictions by all of the methods when two views were missing. Again, GBVS and CVS failed to predict that the cat on the table would be favourably looked at rather than chairs with light shone on them. Both TMVL with H_E and H_{EI} were able to detect the cat; however, TMVL-H_{EI} was able to capture more eye movements than TMVL-H_E did, especially for User 2.

3.2. Fixing a missing view

In this scenario, we wanted to predict where a user would be looking at in a (test) set of images based on where the other seven users were looking at in the same set. According to Tables 3–5, TMVL was the best contender for all users, followed by GBVS and CVS in that order. Using H_{EI} as input features could improve prediction performance significantly for all cases and on all matrices.

Compared to GBVS and CVS, TMVL-H_E showed an improvement with AUROC for all users at 3.01% and 11.20% on the average, respectively, as shown in Fig. 15a. TMVL-H_{EI} performance was improved much more, 5.81% against GBVS and 14.22% against CVS on the average, as shown in Fig. 15b. Our proposed algorithm achieved the highest improvement compared to both baselines for User 2. This means that other users' eye movements behaviours were very useful to User 2. However, the least improvement was found for User 8. This indicates that user adaptation can be useful as the performance was improved even though not so much for the case of User 8. Using TMVL-H_{EI} yielded better AUROC, at 2.73% on the average than TMVL-H_E did, as shown in Fig. 16.

Table 3
A comparison between all of the methods with AUROC when only one fixed view/user are missing. Bold values indicate the best AUROC achieved in each user.

Method	User index							
	1	2	3	4	5	6	7	8
TMVL-H_{EI}	**0.8550**	**0.8631**	**0.7939**	**0.8300**	**0.9010**	**0.8468**	**0.8418**	**0.7901**
TMVL-H_E	0.8349	0.8508	0.7690	0.8079	0.8711	0.8246	0.8184	0.7670
GBVS	0.8040	0.7848	0.7638	0.8028	0.8410	0.7888	0.8025	0.7636
CVS	0.7426	0.7164	0.7146	0.7517	0.7708	0.7257	0.7457	0.7171

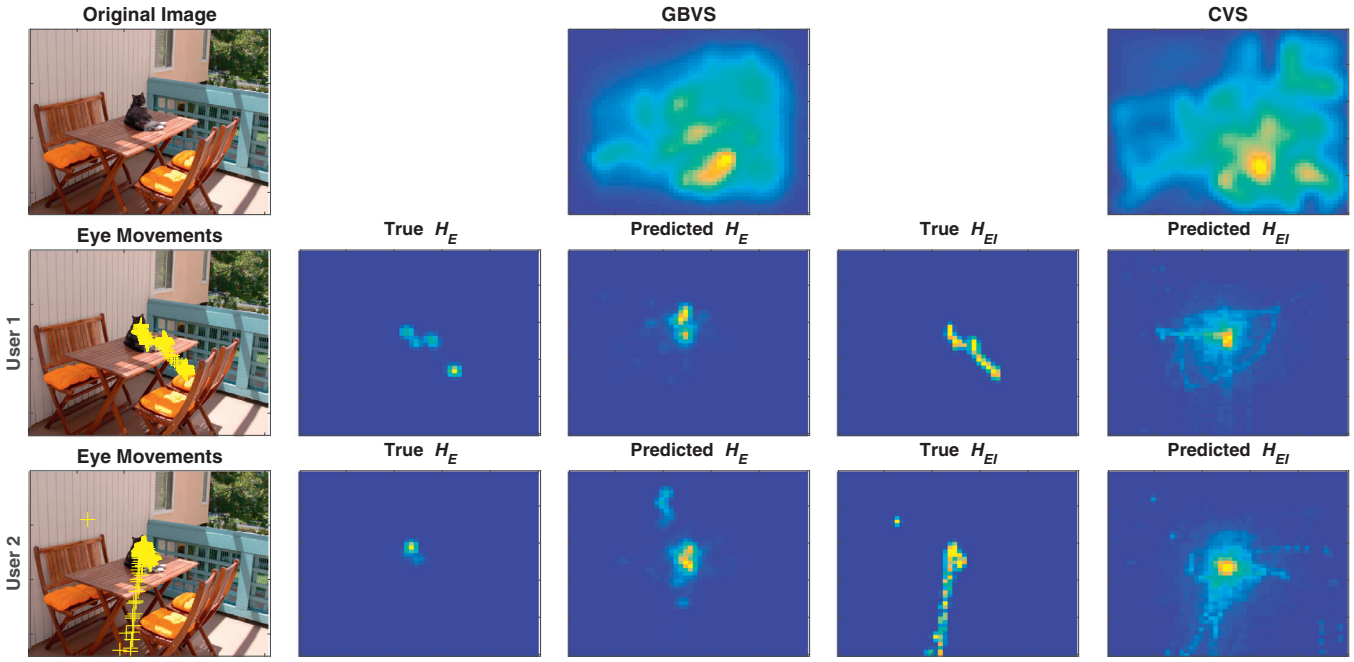


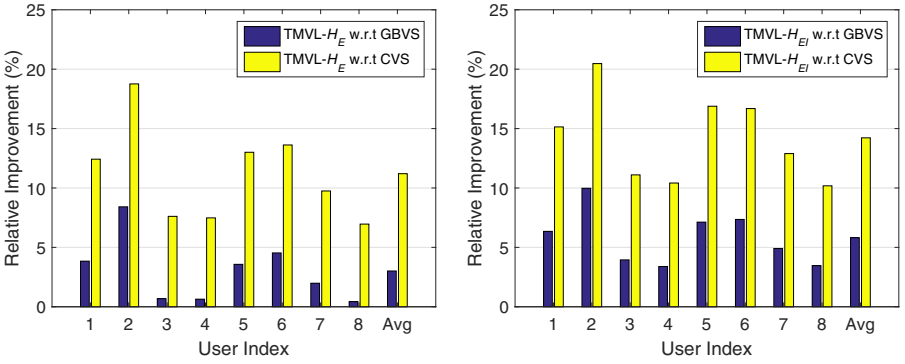
Fig. 14. A comparison of predicted heatmaps by all of the considered methods.

Table 4
A comparison between all of the methods with correlation when only one fixed view/user are missing.

Feature	Method	User index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.4147	0.4747	0.3568	0.4046	0.4569	0.4266	0.4154	0.3657
	GBVS	0.2310	0.2319	0.2356	0.2736	0.2259	0.2432	0.2660	0.2645
	CVS	0.1762	0.1757	0.1829	0.2114	0.1727	0.1829	0.2045	0.2097
H_E	TMVL	0.3276	0.4433	0.2951	0.3437	0.4013	0.3874	0.3706	0.2861
	GBVS	0.1835	0.1932	0.1856	0.2083	0.1928	0.1981	0.2135	0.1976
	CVS	0.1425	0.1465	0.1476	0.1662	0.1501	0.1526	0.1675	0.1604

Table 5
A comparison between all of the methods with JSD when only one fixed view/user are missing.

Feature	Method	User index							
		1	2	3	4	5	6	7	8
H_{EI}	TMVL	0.6324	0.5279	0.5152	0.4866	0.6234	0.5364	0.4998	0.4095
	GBVS	0.7442	0.7046	0.5912	0.5893	0.8102	0.6735	0.6193	0.4921
	CVS	0.7625	0.7171	0.5927	0.5968	0.8343	0.6844	0.6277	0.4833
H_E	TMVL	0.6792	0.5463	0.7011	0.6606	0.6055	0.6083	0.6269	0.7003
	GBVS	1.1585	1.1138	1.0224	1.0024	1.1760	1.0451	0.9996	0.9582
	CVS	1.2184	1.1753	1.0757	1.0595	1.2376	1.1056	1.0587	1.0103



(a) H_E (b) H_{EI}

Fig. 15. Relative improvement of TMVL compared to GBVS and CVS when only one fixed view/user was missing.

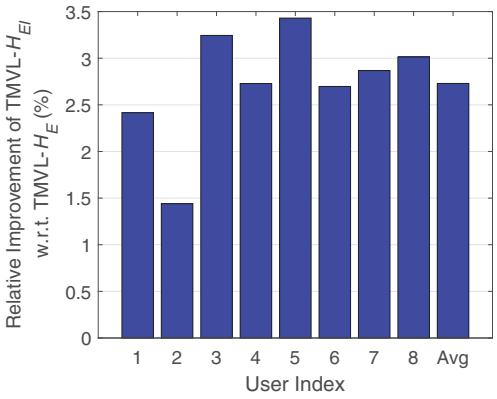


Fig. 16. Relative improvement of TMVL-H_{EI} compared to TMVL-H_E when only one fixed view/user was missing.

4. Conclusion

In this paper, we introduced a missing-value prediction schema built upon maximum-margin-based learning and invariances of tensor algebra. Our proposed algorithm was tested on an eye movement dataset in order to identify where users were looking at in images. We also proposed a new technique that decomposes images called "Kronecker Decomposition". This technique can be used in image compression applications. We also proposed an approach to combine image with eye movement data. The processed image is then decomposed by using Kronecker decomposition. We have demonstrated that our framework was able to perform better than two well-known saliency detection techniques. Several initial results show that user adaptation may be useful; thus, user information should be investigated further in future research.

Acknowledgement

This work was supported by the Thailand Research Fund under grant agreement number TRG5680090.

References

[1] L. Itti, C. Koch, E. Niebur, A model of saliency-based visual attention for rapid scene analysis, IEEE Trans. Pattern Anal. Mach. Intell. 20 (11) (1998) 1254–1259, doi:10.1109/34.730558.
[2] J. Harel, C. Koch, P. Perona, Graph-based visual saliency, in: Advances in Neural Information Processing Systems, 2006, pp. 545–552.
[3] A. Borji, M.-M. Cheng, H. Jiang, J. Li, Salient object detection: A survey, arXiv:1411.5878(2014).
[4] T. Judd, K. Ehinger, F. Durand, A. Torralba, Learning to predict where humans look, in: Proceeding of the IEEE 12th International Conference on Computer Vision, 2009, pp. 2106–2113, doi:10.1109/ICCV.2009.5459462.
[5] J.M. Henderson, J.R. Brockmole, M.S. Castelhano, M. Mack, Visual saliency does not account for eye movements during visual search in real-world scenes, Eye Mov. Window Mind Brain (2007) 537–562, doi:10.1016/B978-008044980-7/50027-6.
[6] Q. Zhao, C. Koch, Learning a saliency map using fixated locations in natural scenes, J. Vis. 11 (3) (2011) 1–15, doi:10.1167/11.3.9.
[7] M. Liang, X. Hu, Feature selection in supervised saliency prediction, IEEE Trans. Cybern. 45 (5) (2015) 914–926, doi:10.1109/TCYB.2014.2338893.
[8] J. Wang, A. Borji, C.J. Kuo, L. Itti, Learning a combined model of visual saliency for fixation prediction, IEEE Trans. Image Process. 25 (4) (2016) 1566–1579, doi:10.1109/TIP.2016.2522380.
[9] L. Zhang, X. Li, L. Nie, Y. Yang, Y. Xia, Weakly supervised human fixations prediction, IEEE Trans. Cybern. 46 (1) (2016) 258–269, doi:10.1109/TCYB.2015.2400821.
[10] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 208–220, doi:10.1109/TPAMI.2012.39.
[11] C.-Y. Chen, K. Grauman, Inferring unseen views of people, in: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2011–2018, doi:10.1109/CVPR.2014.258.
[12] D.R. Hardoon, K. Pasupa, J. Shawe-Taylor, Image ranking with implicit feedback from eye movements, in: C.H. Morimoto, H.O. Istance, A. Hyrskykari, Q. Ji (Eds.), Proceeding of the 6th Biennial Symposium on Eye Tracking Research & Applications (ETRA 2010), 22–24 March 2010, Austin, USA, 2010, pp. 291–298, doi:10.1145/1743666.1743734.
[13] P. Auer, Z. Hussain, S. Kaski, A. Klami, J. Kujala, J. Laaksonen, A.P. Leung, K. Pasupa, J. Shawe-Taylor, Pinview: implicit feedback in content-based image retrieval, in: T. Diethe, N. Cristianini, J. Shawe-Taylor (Eds.), Proceeding of the Workshop on Applications of Pattern Analysis, Journal of Machine Learning Research - Proceedings Track, 11, Cumberland Lodge, UK, 2010, pp. 51–57.
[14] Z. Hussain, A.P. Leung, K. Pasupa, D.R. Hardoon, P. Auer, J. Shawe-Taylor, Exploration-exploitation of eye movement enriched multiple feature spaces for content-based image retrieval, in: J.L. Balcázar, F. Bonchi, A. Gionis, M. Sebag (Eds.), Proceeding of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Part I, Lecture Notes in Computer Science, 6321, Barcelona, Spain, 2010, pp. 554–569, doi:10.1007/978-3-642-15880-3_41.
[15] K. Pasupa, P. Chatkamjuncharoen, C. Wuttilertdeshar, M. Sugimoto, Using image features and eye tracking device to predict human emotions toward abstract images, in: T. BrÃ;unl, B. McCane, M. Rivers, X. Yu (Eds.), Proceeding of the 7th Pacific Rim Symposium on Image and Video Technology, Lecture Notes in Computer Science, 9431, Auckland, New Zealand, 2016, pp. 419–430, doi:10.1007/978-3-319-29451-3_34.
[16] M. Itskov, Tensor Algebra and Tensor Analysis for Engineers With Applications to Continuum Mechanics, second, Springer, 2009.
[17] J. Synge, A. Schild, Tensor Calculus, Dover, 1978.
[18] K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, J. Rousu, Towards structured output prediction of enzyme function, in: BMC Proceedings, 2, Suppl 4, 2008, p. S2, doi:10.1186/1753-6561-2-s4-s2.
[19] S. Szedmak, T. De Bie, D.R. Hardoon, A metamorphosis of canonical correlation analysis into multivariate maximum margin learning, in: Proceeding of the 15th European Symposium on Artificial Neural Networks, 2007, pp. 211–216.
[20] T. Joachims, Training linear SVMs in linear time, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA, 2006, pp. 217–226, doi:10.1145/1150402.1150429.
[21] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.
[22] C.V. Loan, The ubiquitous Kronecker product, J. Comput. Appl. Math. 123 (2000) 85–100. The nearest Kronecker product
[23] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[24] L. Pachter, B. Sturmfels, Algebraic Statistics for Computational Biology, Cambridge University Press, 2005.
[25] M. Drton, B. Sturmfels, S. Sullivant, Lectures on Algebraic Statistics, Oberwolfach Seminars, vol 40, Birkhauser, 2009.
[26] D.G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the 7th IEEE International Conference on Computer Vision, 2, 1999, pp. 1150–1157, doi:10.1109/ICCV.1999.790410.
[27] C. Schmid, R. Mohr, C. Bauckhage, Evaluation of interest point detectors, Int. J. Comput. Vis. 37 (2) (2000) 151–172, doi:10.1023/A:1008199403446.
[28] E.-C. Chang, S. Mallat, C. Yap, Wavelet foveation, Appl. Comput. Harmon. Anal. 9 (3) (2000) 312–335.
[29] J. Briët, P. Harremoës, Properties of classical and quantum Jensen–Shannon divergence, Phys. Rev. A 79 (5) (2009) 052311.
[30] K. Pasupa, S. Szedmak, Learning to predict where people look with tensor-based multiview learning, in: S. Arik, T. Huang, W.K. Lai, Q. Liu (Eds.), Proceeding of the 22nd International Conference on Neural Information Processing, Lecture Notes in Computer Science, 9489, Istanbul, Turkey, 2015, pp. 432–441, doi:10.1007/978-3-319-26532-2_47.

Kitsuchart Pasupa is an Assistant Professor in the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. Previously, he was a research fellow at Southampton and Sheffield Universities. He obtained a BEng in Electrical Engineering from Sirindhorn International Institute of Technology, Thammasat University, Thailand in 2003, and then went to the Department of Automatic Control and Systems Engineering, University of Sheffield where he received his MSc(Eng) and PhD in Automatic Control and Systems Engineering in 2004 and 2008, respectively. His main research interests lie in the application of machine learning techniques in the real world application.

Sandor Szedmak was born in Hungary. He obtained an MS degree in Mathematics from Lajos Kossuth University, Debrecen, Hungary. He received his PhD in Operations Research from the Rutgers, the State University of New Jersey, in the USA. He held research positions at the Computer Science Department of Royal Holloway, University of London and the University of Helsinki. Following that, he was a senior research fellow at the School of Electronics and Computer Science of University of Southampton in the UK, and in the Informatics Institute at the University of Innsbruck in Austria. Recently, he is a senior researcher at the Department of Computer Science of the Aalto University in Finland. His main research interest focuses on mathematical problems, e.g. optimisation, arising in machine learning and pattern recognition.