Universitat Politècnica de Catalunya

MIRI-KMLMM

k-KISSME

Authors:
Julian Kunze
Miro Pütz
Sonja Ginter

January 9, 2019

Contents

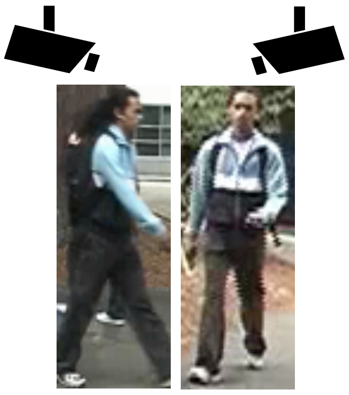
Figure 1: Identical instance captured from the two cameras from different viewpoints in the VIPeR dataset

1 Introduction

In this work we want to solve a problem regarding video surveillance and content-based image retrieval and a wide range of other applications. Given a query image, the goal is to retrieve all images that have the same identity from a large gallery. The main issue to overcome is that query and gallery image are often taken from different viewpoints and with different cameras under varying illumination conditions (Fig. 1). Problems also arise from different backgrounds, resolution, pose, occlusion and similar clothing. Because of that it is a challenging task, even for humans. It is common to measure the performance of an algorithm in re-identification tasks by a top rank k matching rate. This expresses the percentage of query images with correct matches found in the top-k ranked gallery images. More simply, a correct match in the first top 10 or 20 ranks is often sufficient.

Different approaches in re-identification have been tested, which in general try to find a good feature representation of the images and a distance metric to discriminate between them. In most studies the features are either extracted by handcrafted descriptors or by using deep convolutional neural networks (DCNNs) [1, 2], which have achieved remarkable results in many image recognition challenges. The main drawback of using DCNNs is the lack of sufficient training data. Many datasets provide only two images for each identity such as VIPeR [3], which we will use in this study. Therefore, the siamese model is often used in re-identification, which is a special kind of neural network. It is popular among tasks that involve finding similarities between objects [4, 1].

Regarding the distance metric, it should reflect the true underlying relationship between images to generalize well to unseen data. To improve the performance, additional information in form of labeled data supports the distance metric. However, collecting a fully labeled dataset with class labels is very demanding and not really necessary. It is much simpler and efficient to provide labels in form of must-link constraints, e.g. images of the same person, are used to specify that the examples belong to the same class. On the other hand, cannot-link constraints denote pairs of images that do not belong to the same class. The benefit of this approach is that a user does not have to think about the exact label of an image. It is sufficient to ask a user whether two images display the same person or not. The goal of the distance metric should be to minimize the distances between must-link constraints, while the distance between cannot-link constraints should be large. This optimization problem of semidefinite programming becomes intractable on a large scale because of the computational complexity. Köstinger et al. [5] proposed a distance metric learning approach (named KISSME) that is very efficient to obtain and is motivated by a statistical inference perspective based on a likelihood-ratio test. Therefore, the method is scalable to large datasets, as it just involves computation of two small sized covariance matrices.

Based on KISSME, Nguyen and De Baets [6] derived a kernelized distance metric (named k-KISSME), which enables capturing the nonlinear structure in a dataset. The goal of our study is to implement the proposed k-KISSME method. We expect to get a computationally efficient and robust approach for the person re-identification task. Furthermore, we compare our results with those of the papers and try to reproduce the same performance.

In the next Section, we are reviewing the theoretical foundation regarding our applied feature extraction (2.1), the ordinary KISSME method (2.2) and its kernelized successor (2.3). Afterwards, we are explaining our implementation (3) and the tests we performed to validate the algorithm (4). We conclude our report discussing the results (5) and giving an outlook to further research (6).

2 Theory

As already stated in Section 1 most methods for re-identification tasks are divided into two parts: finding a suitable feature representation and learning a good metric to discriminate between particular instances. Here in this section, we will lay the theoretical foundations of our work by first explaining the feature extraction method (Section 2.1), subsequently describing the ordinary KISSME method (Section 2.2) and finally deriving the kernelized version of this algorithm (Section 2.3).

2.1 Feature Extraction

In previous work, Liao et al. [7] developed a feature extraction method called Local Maximal Occurrence (LOMO) when also proposing a metric learning technique for person re-identification themselves. It derives both color and texture features from images. The extraction is designed in a way so that illumination and viewpoint

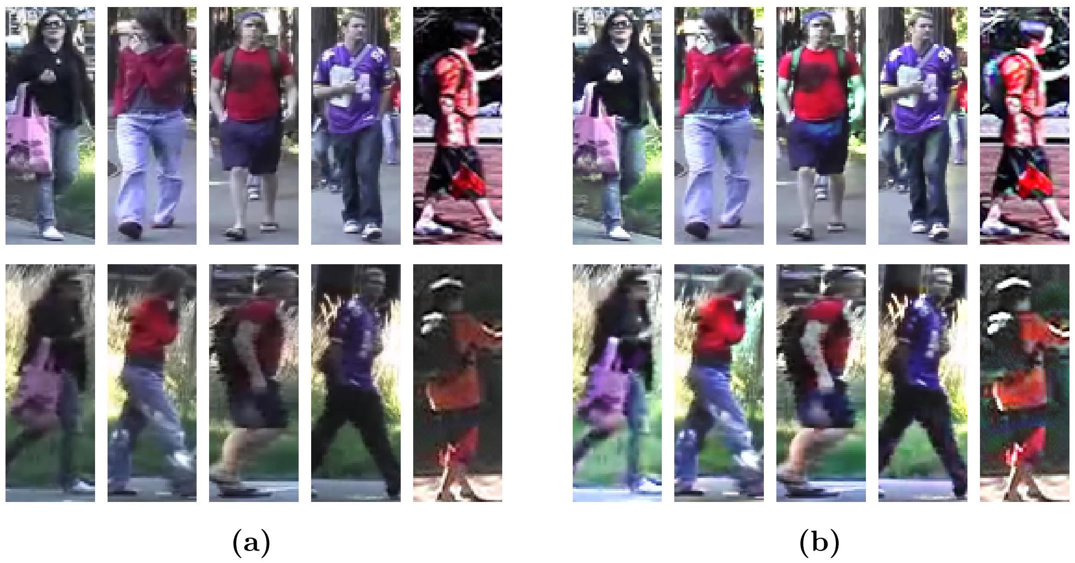

Figure 2: Showcase images from the VIPeR dataset [3] (compilation from [7]), where each column displays the same person. (a) original images, (b) processed images (Retinex algorithm)

variations have a minimal influence on the features which are fundamental challenges in re-identification tasks.

In order to tackle changes in illumination that occur due to image matching between different cameras at varying locations, the Retinex algorithm [8, 9], more precisely the multiscale Retinex transformation by Jobson et al. [10] is applied. The method aims at processing images in a way that they approximate the human visual perception thus they become more invariant in lighting and color. Figure 2 contrasts images from the VIPeR dataset [3] (Fig. 2a) with their processed counterparts (Fig. 2b). In general, after processing the images' colors appear more vivid and consistent. Accordingly, color features are extracted in the following way:

1. The Retinex algorithm is applied to become more independent of illumination.
2. The images RGB values are transformed into the HSV color space.
3. Based on the HSV color space a histogram is generated where the bin frequencies are used as features

The LOMO method subdivides each scale of the color space into eight bins so that the whole histogram consists of $8 \times 8 \times 8 = 512$ bins.

The texture feature set is generated applying the so-called Scale Invariant Local Ternary Pattern (SILTP) [11] descriptor which describes the image's local textures. As opposed to the color features, it is not applied to the Retinex processed but the original image. Furthermore, it differs from the Local Binary Pattern (LBP) [12]

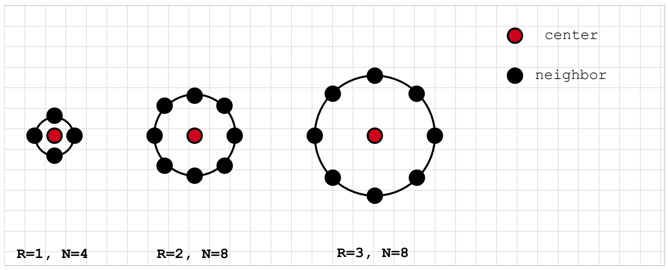

Figure 3: Examples of neighborhoods used for the computation of SILTP features (The grid granularity is on pixel-level)

descriptor, as SILTP is invariant to intensity scale changes and robust against image noises. The descriptor is generated as follows:

1. The image is transformed into grayscale.
2. For every pixel c in the image (described by its location x_c, y_c) the binary string defined by

$$SILTP^{\tau}_{N,R}\left(x_c, y_c\right) = \bigoplus_{k=0}^{N-1} s_{\tau}\left(I_c, I_k\right)$$

   is computed where I_c defines the grayscale intensity of pixel c and I_k expresses the respective grayscale intensity of the kth of N neighboring pixels that are equally distributed on a circle of radius R around the center pixel c (see Fig. 3). Furthermore, s_τ describes the function for comparing the center pixel with one of its neighbors and is defined as

$$s_{\tau}\left(I_c, I_k\right) = \begin{cases} 01, & \text{if } I_k > (1+\tau)I_c \\ 10, & \text{if } I_k < (1-\tau)I_c \\ 00, & \text{otherwise.} \end{cases}$$

   Finally, $\bigoplus$ denotes an operator for concatenating the binary strings returned by s_τ and τ describes the scaling factor that indicates the range of comparison performed by s_τ.
3. Based on the binary string for each pixel, a histogram is computed with a bin for each different binary string.

This histogram is then used for the SILTP features. Since there are three possible binary strings for each neighbor, every pixel can be one of 3^N different binary strings which results in 3^N bins in the histogram and thus 3^N features. The property that SILTP is more invariant to intensity scale changes and more robust against image noises than LBP can be easily seen in Figure 4.

In order to achieve robustness against viewpoint changes, both the HSV color histogram and the SILTP histogram are not applied to the full image but instead the

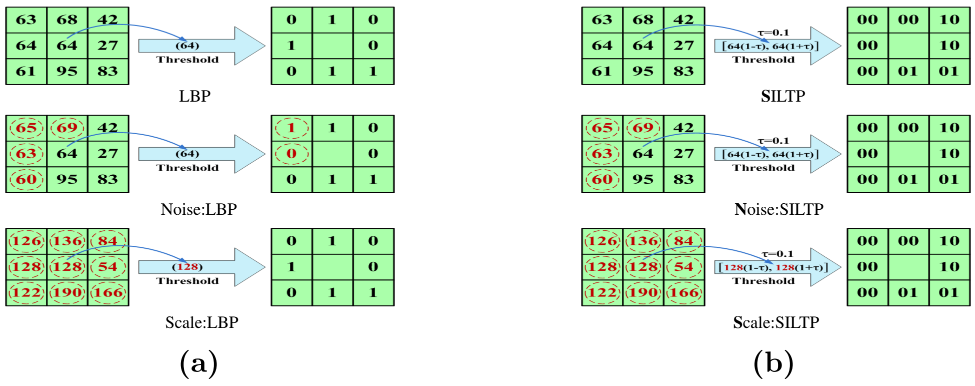

Figure 4: Comparison between LBP (a) and SILTP (b). "First row: original encodings. Second row: encodings with noises. Third row: encodings with scale transform (all pixel values are doubled). The circled red pixels are changed with noises or by scale transform, and the circled red encodings are affected by those changes correspondingly." [11]

image is subdivided into smaller parts by a sliding window approach. For obtaining the LOMO features a subwindow of size 10×10 is used that overlaps by 5 pixels in every step. In each of these subwindows the $8 \times 8 \times 8$ HSV color histogram and two SILTP histograms are computed. Both SILTP histograms are parameterized by $\tau = 0.3$ and $N = 4$ but they differ in their applied circle radius, where one uses $R = 3$ and the other $R = 5$. Afterwards, all subwindows on the same horizontal line are merged as for every bin in all three histograms only the maximum value is kept (hence the name Local Maximal Occurrence). This procedure is graphically displayed in Figure 5. According to Liao et al. [7] the result is "some invariance to viewpoint changes".
Furthermore, not only a feature vector is generated for the original size image but also for two downsampled versions. Thereby, multi-scale information is further considered. The downsampling is realized by applying a 2×2 average pooling operation. Since the LOMO method extracts histograms for every row of subwindows, the number of horizontal groups of subwindows determines the total number of features. This number n_h is dependent on the height h of the image, the subwindow size $s_{subwindow}$ and the step size s_{step}:

$$n_h = \lfloor \frac{h - (s_{subwindow} - s_{step})}{s_{step}} \rfloor$$

For images from the VIPeR dataset that are 48×128 pixels large this results in 24 rows for the original image plus 11 and 5 rows for the downsampled images.

The final descriptor is then generated by concatenating all of the local maximal occurrences. This yields a feature vector of

$$\left(8 \times 8 \times 8 \text{ color bins} + 3^4 \cdot 2 \text{ SILTP bins}\right) \cdot (24 + 11 + 5 \text{ horizontal groups}) = 26960$$

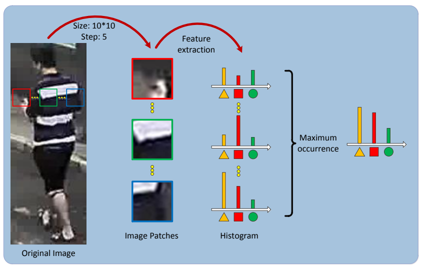

Figure 5: Addressing viewpoint changes by keeping only the maximal occurrence of each pattern for all subwindows that share the same horizonal position

dimensions. As a very last step, large bin values are suppressed by applying a log transform to all features and both HSV features and SILTP feature vectors are individually normalized to unit length.

2.2 KISSME

KISSME is the short name for the **K**eep **I**t **S**imple and **S**traightforward **ME**tric proposed by Köstinger et al. [5]. The idea is to re-identify an object or person by pairwise comparing the image data (given as extracted features). In doing so, the method follows a two-track approach. It generates the observed commonalities of similar and dissimilar pairs. Through a likelihood ratio test it is possible to make statistical decisions if a pair is similar or dissimilar. For this purpose, two hypotheses are set up:

$$\begin{aligned} H_0 &= \text{for comparing the dissimilarity} \\ H_1 &= \text{for the similarity.} \end{aligned}$$

The log-likelihood ratio is applied to examine whether $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class or not

$$\sigma(\mathbf{x}_i, \mathbf{x}_j) = \log\left(\frac{p(\mathbf{x}_i, \mathbf{x}_j | H_0)}{p(\mathbf{x}_i, \mathbf{x}_j | H_1)}\right) \tag{1}$$

where p denotes the probability density function. A high value $\sigma(\mathbf{x}_i, \mathbf{x}_j)$ indicates a dissimilar pair and H_0 holds true. The other way around, if the value is low H_0 is rejected and thus the pair is classified as similar [5]. Transforming the data into a space of the differences $(\mathbf{x}_i - \mathbf{x}_j)$ that is independent of locality and assuming a normal distribution with zero mean for these differences [6] we get

$$\sigma(\mathbf{x}_i,\mathbf{x}_j) = \log\left(\frac{p(\mathbf{x}_i-\mathbf{x}_j|H_0)}{p(\mathbf{x}_i-\mathbf{x}_j|H_1)}\right)$$
$$= \log\left(\frac{\frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_0|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^\top\mathbf{\Sigma}_0^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right)}{\frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}_1|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^\top\mathbf{\Sigma}_1^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right)}\right)$$

where $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$ are the respective covariance matrices of $p(\mathbf{x}_i-\mathbf{x}_j|H_0)$ and $p(\mathbf{x}_i-\mathbf{x}_j|H_1)$. Simplifying the formula results in

$$\sigma(\mathbf{x}_i,\mathbf{x}_j) = \frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{\Sigma}_1^{-1}-\mathbf{\Sigma}_0^{-1})(\mathbf{x}_i-\mathbf{x}_j) + log\left(\frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_0|}\right) \tag{2}$$

where constant terms can be neglected since they do not affect the log-likelihood:

$$= (\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{\Sigma}_1^{-1}-\mathbf{\Sigma}_0^{-1})(\mathbf{x}_i-\mathbf{x}_j) \tag{3}$$

This way, we get an expression representing the Mahalanobis distance so that we only need to estimate both covariance matrices in order to obtain a meaningful metric.

Mahalanobis Distance The Mahalanobis distance measures the squared distance between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ of the samples i and j rectified by the covariance matrix $\mathbf{M}$ [5].

$$d_M^2(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{M}(\mathbf{x}_i-\mathbf{x}_j) \tag{4}$$

with $\mathbf{M} \geq 0$ and PSD, $\mathbf{x}_i,\mathbf{x}_j \in \mathbb{R}^d$.

The covariance matrices are estimated with the maximum likelihood as follows [6, eq.4]:

$$\mathbf{\Sigma}_0 = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T \tag{5}$$

$$\mathbf{\Sigma}_1 = \frac{1}{|\mathcal{S}|}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T \tag{6}$$

where $\mathcal{D}$ denotes a set of image pairs where both images of the pair belong to the same instance and $\mathcal{S}$ respectively classifies a set with pairs from different instances. Nguyen and De Baets [6] call $\mathcal{D}$ the set of must-link constraints and $\mathcal{S}$ the set of cannot-link constraints.

2.3 kernelized KISSME

In this section we try to overcome some of the limitations of KISSME. Because it uses the features to calculate the covariance matrices $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$, a high number of features (D) is a limitation for KISSME: The high memory complexity $\mathcal{O}(D^2)$ for computing the covariance together with calculating the inverse becomes intractable for high-dimensional data. In addition, it is possible that the covariance matrix becomes singular and therefore not invertible. We utilize the kernel method to efficiently calculate the covariance matrices. In contrast to KISSME the kernel method makes the calculation of the covariance matrices independent of the number of features.

In order to kernelize KISSME a suitable representation of the covariance matrices has to be found, where operations on the data only appear as inner products. The inner product can then be replaced by a kernel function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is able to implicitly calculate the inner product in a high-dimensional feature space. With the kernel function k and inputs $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathcal{X}$, the $n \times n$ Gram matrix (or kernel matrix) $\mathbf{K} := (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ can be constructed.

The Gram matrix can be computed without carrying out the explicit map $K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, where ϕ is the nonlinear map from the input space $\mathcal{X}$ into a high-dimensional feature space $\mathcal{F}$, i.e. $\phi\colon \mathcal{X} \to \mathcal{F}$. Instead, kernel functions are used for example the radial basis function. This is called the kernel trick.

The following derivation of a kernel distance metric follows exactly the paper of Nguyen and De Baets [6]. For finding a representation that is suitable for a kernelized version we exploit the fact that the constraints of must-and cannot-links are given. Let $\mathcal{S}$ denote the set of must-links and $\mathcal{D}$ the set of cannot-links. First we introduce some notation and define some new matrices. Let $\mathbf{1}_i$ be a column vector that has the value 1 at the i -th entry and zeros at all other entries. Additionally, let $\mathbf{B}_0$ (resp. $\mathbf{B}_1$) be a $n \times n$ diagonal matrix whose diagonal vector contains at the i-th entry the number of constraints in $\mathcal{D}$ (resp. $\mathcal{S}$) of which the first element is $\mathbf{x}_i$,i.e.

$$\begin{aligned} \operatorname{diag}(\mathbf{B}_0)_i) &= |\{j \mid j \in \{1..., n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \text{ in } \mathcal{D}\}| \\ \operatorname{diag}(\mathbf{B}_1)_i) &= |\{j \mid j \in \{1..., n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \text{ in } \mathcal{S}\}| \end{aligned}$$

Finally, let $\mathbf{E}_0$ (resp. $\mathbf{E}_1$) be a $n \times n$ diagonal matrix whose diagonal vector contains at the j-th entry the number of constraints in D (resp. S) of which the second element is $\mathbf{x}_j$, i.e.

$$\begin{aligned} \operatorname{diag}(\mathbf{E}_0)_j) &= |\{i \mid i \in \{1..., n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \text{ in } \mathcal{D}\}| \\ \operatorname{diag}(\mathbf{E}_1)_j) &= |\{i \mid i \in \{1..., n\} \text{ and } (\mathbf{x}_i, \mathbf{x}_j) \text{ in } \mathcal{S}\}| \end{aligned}$$

Let $\mathbf{W}_0$ (resp. $\mathbf{W}_1$) be an $n \times n$ matrix whose entry at the i-th row and j-th column is 1 if $(\mathbf{x}_i,\ \mathbf{x}_j) \in \mathcal{D}$ (resp. $(\mathbf{x}_i,\ \mathbf{x}_j) \in \mathcal{S}$), otherwise it takes value 0. Using the preceding notations, we can rewrite the covariance matrix $\mathbf{\Sigma}_0$ in eq. 7 as

$$\begin{aligned}\mathbf{\Sigma}_0 &= \frac{1}{n_0}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{x}_i\mathbf{x}_i^T - \mathbf{x}_j\mathbf{x}_i^T - \mathbf{x}_i\mathbf{x}_j^T + \mathbf{x}_j\mathbf{x}_j^T)\\ &= \frac{1}{n_0}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{X}\mathbf{1}_i\mathbf{1}_i^T\mathbf{X}^T - \mathbf{X}\mathbf{1}_j\mathbf{1}_i^T\mathbf{X}^T - \mathbf{X}\mathbf{1}_i\mathbf{1}_j^T\mathbf{X}^T + \mathbf{X}\mathbf{1}_j\mathbf{1}_j^T\mathbf{X}^T)\\ &= \frac{1}{n_0}\mathbf{X}(\mathbf{B}_0 - \mathbf{W}_0^T - \mathbf{W}_0 + \mathbf{E}_0)\mathbf{X}^T\\ &= \frac{1}{n_0}\mathbf{X}\mathbf{H}_0\mathbf{X}^T,\end{aligned} \tag{7}$$

where $\mathbf{H}_0 = \mathbf{B}_0$ - $\mathbf{W}_0^T$ - $\mathbf{W}_0$ + $\mathbf{E}_0$.
For $\mathbf{\Sigma}_1$ the same formula applies just that every zero-index subscript is replaced by a one. The covariance matrix can still be singular due to the lack of sufficient pairwise constraints. To prevent this problem a small positive constant value ϵ is added to the diagonals of $\mathbf{\Sigma}_0$ and $\mathbf{\Sigma}_1$, i.e.

$$\widehat{\mathbf{\Sigma}}_0 = \epsilon\mathbf{I} + \frac{1}{n_0}\mathbf{X}\mathbf{H}_0\mathbf{X}^T, \qquad \widehat{\mathbf{\Sigma}}_1 = \epsilon\mathbf{I} + \frac{1}{n_1}\mathbf{X}\mathbf{H}_1\mathbf{X}^T \tag{8}$$

To evaluate the inverses of these matrices, we consider the Woodbury matrix identity given by

$$(\mathbf{A}+\mathbf{B}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I}+\mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \tag{9}$$

Applying eq.(9) results in the following form for the covariances

$$\widehat{\mathbf{\Sigma}}_0^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n_0\epsilon^2}\mathbf{X}\mathbf{H}_0(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{X}^T\mathbf{X}\mathbf{H}_0)^{-1}\mathbf{X}^T \tag{10}$$

$$\widehat{\mathbf{\Sigma}}_1^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{n_1\epsilon^2}\mathbf{X}\mathbf{H}_1(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{X}^T\mathbf{X}\mathbf{H}_1)^{-1}\mathbf{X}^T \tag{11}$$

Finally in eq. 10 and 11 we can replace $\mathbf{X}^T\mathbf{X}$ by a $n \times n$ kernel matrix $\mathbf{K}$. Moreover, we can then also insert the difference of covariances into the Mahalanobis distance formula from eq.(3). With this expression for our covariances, $\mathbf{M}$ can be computed as

$$\begin{aligned}\mathbf{M} = \widehat{\mathbf{\Sigma}}_1^{-1} - \widehat{\mathbf{\Sigma}}_0^{-1} &= \mathbf{X}\Bigg[\frac{1}{n_0\epsilon^2}\mathbf{X}\mathbf{H}_0(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{K}\mathbf{H}_0)^{-1}\\ &\qquad - \frac{1}{n_1\epsilon^2}\mathbf{X}\mathbf{H}_1(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{K}\mathbf{H}_1)^{-1}\Bigg]\mathbf{X}^T\\ &= \mathbf{X}\mathbf{C}\mathbf{X}^T,\end{aligned} \tag{12}$$

where **C** denotes the term in the square brackets. With this the Mahalanobis distance evaluates to

$$\begin{aligned} d^2_{\mathbf{M}}(\mathbf{x}_i \, \mathbf{x}_j) &= (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{X}\mathbf{C}\mathbf{X}^T(\mathbf{x}_i - \mathbf{x}_j) \\ &= (\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j})^T \mathbf{C}(\mathbf{k}_{\mathbf{x}_i} - \mathbf{k}_{\mathbf{x}_j}) \end{aligned} \tag{13}$$

where $k_{\mathbf{x}} = \mathbf{X}^T\mathbf{x}$.
This kernelized notation allows us to replace inner products between samples by a kernel function to perform distance metric learning in the feature space. In addition, to the benefits of the nonlinearity of the kernel it allows us to apply the distance metric on data sets containing structured objects on which kernel functions are defined.
As mentioned before, the computational complexity of this kernelized version is not dependent on the number of features anymore. Instead, the calculation mainly scales with the calculation of **C**. Due to matrix multiplications and inversions, this computation scales as $\mathcal{O}(n^3)$. The implication is an improved performance for problems where the number of features is significantly larger than the number of examples, i.e. $D \gg n$ [6].

3 Implementation

Our implementation is roughly divided into two parts: The LOMO feature extraction and the kernelized KISSME. Although the feature extraction is a substantial part to make the metric learning applicable, we did not bother much implementing the feature extraction since our work's focus lies on the kernelized KISSME algorithm. Instead we used an existing solution from Github[1] where a LOMO feature extractor has been implemented in Python.

After comparing this implementation with the originally characterized method by Liao et al. [7] and verifying its correctness we adapted it to our needs. We changed some commands to conform with Python version 3 and added code for importing the images and writing the LOMO features into a data file that can then be accessed by the metric learning implementation. Furthermore, we commented the whole code to make it better comprehensible as it was left completely uncommented by the creator. The feature extractor can be run by simply executing the `run.py` file. It then processes all datasets from the given directory and creates a data file for every dataset, containing the feature vectors of all the images of that dataset. In our case, it processes the images of the VIPeR dataset and creates one file with an image for every of the 632 subjects and another file with an image taken from a different angle also for all 632 pedestrians.

[1]`https://github.com/dongb5/LOMO-feature-extractor`

Our kernelized KISSME algorithm then imports these features to learn the resultant distance metric. As a first step, the dataset is split up into training, validation and test data. The training and validation data is used to learn the model and optimize the model's hyperparameters (see Section 4). The test data is used at the very end to report our final model's performance. Must- and cannot-link constraints are generated only based on the training instances: Determining a must-link for every image in the dataset is easily accomplished as images of the same person hold the same index in both camera viewpoint sets. That is why the must links are expressed as a data frame containing two columns where the values in each row are the same representing the training images. In order to have a balanced number of must- and cannot-link constraints for every image of camera set A only one cannot-link constraint is built. They are constructed assigning every image from set A randomly another image from set B that shows a different pedestrian. The pairs of images are then again stored in a data frame and as a result there is one must- and one cannot-link constraint for every image of set A.
As a next step the kernel matrix **K** is computed dependent on which kernel method is used (see Section 4). Based on this kernel matrix the **C** matrix from eq. 12 can be computed.

$$\begin{aligned}\mathbf{C} = &\frac{1}{n_0\epsilon^2}\mathbf{X}\mathbf{H}_0(\mathbf{I} + \frac{1}{n_0\epsilon}\mathbf{K}\mathbf{H}_0)^{-1}\\ &- \frac{1}{n_1\epsilon^2}\mathbf{X}\mathbf{H}_1(\mathbf{I} + \frac{1}{n_1\epsilon}\mathbf{K}\mathbf{H}_1)^{-1}\end{aligned}$$

Both **H** matrices are computed first. They are sparse because they only contain $3 \cdot n_c$ values in a $n \times n$ matrix due to the way they are defined where $n_c = r_{train} \cdot n$. Here, n denotes the number of pedestrians in the dataset, n_c the number of generated constraints (either must- or cannot-link) and r_{train} the ratio of training samples. By making the matrices of the sparse matrix datatype time and memory efficiency can be improved. Furthermore, the matrix $\mathbf{I} + \frac{1}{n\epsilon}\mathbf{K}\mathbf{H}$ of which the inverse is computed and the resulting **C** matrix remain relatively sparse, so they are also forced into the sparse matrix type as well to further save memory and accelerate computation.

Finally, the pairwise distance between images from camera set A and camera set B can be computed using the Mahalanobis distance. It is implemented efficiently by using only matrix operations and computing it for all images simultaneously. Given the formula for the Mahalanobis distance (eq. 13) it can be rewritten as follows:

$$d_M^2(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{k}_{x_i}^T\mathbf{C}^{-1}\mathbf{k}_{x_i} + \mathbf{k}_{x_j}^T\mathbf{C}^{-1}\mathbf{k}_{x_j} - 2\mathbf{k}_{x_i}^T\mathbf{C}^{-1}\mathbf{k}_{x_j}$$

To keep the notation less overloaded from hereon for every column vector of the kernel matrix **K** the following notation is used $\mathbf{k}_i := \mathbf{k}_{x_i}$.

$$d_M^2(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{k}_i^T\mathbf{C}^{-1}\mathbf{k}_i + \mathbf{k}_j^T\mathbf{C}^{-1}\mathbf{k}_j - 2\mathbf{k}_i^T\mathbf{C}^{-1}\mathbf{k}_j$$

From this form it can be easily seen that the first summand is independent of $\mathbf{k}_j$ whereas the second summand is independent of $\mathbf{k}_i$. Hence, for all distances with the same $\mathbf{k}_i$ the first summand remains equal and likewise for those with the same $\mathbf{k}_j$, so does the second summand. In our case $\mathbf{k}_i$ represents the column vector of an image from camera set A and $\mathbf{k}_j$ that from camera set B, both taken from the kernel matrix. Therefore, we compute the first summand for all instances in camera set A and the second summand for all instances in camera set B. The following computation shows the process only for one set since the computation is identical for both. Writing out the vector-matrix-vector product into sums of scalar multiplications one obtains

$$\mathbf{k}_i^T\mathbf{C}^{-1}\mathbf{k}_i = \sum_{k=1}^{n} \mathbf{k}_i^{(k)} \left(\sum_{l=1}^{n} \mathbf{C}_{kl}^{-1}\mathbf{k}_i^{(l)} \right)$$

where $k_i^{(k)}$ indicates the k-th element of vector k_i and n is the number of rows in $\mathbf{K}$. Given that $\sum_{l=1}^{n} \mathbf{C}_{kl}^{-1}\mathbf{k}_i^{(l)}$ simply denotes the k-th element of the vector that results from the multiplication of $\mathbf{C}^{-1}$ and $\mathbf{k}_i$ this term can be replaced to simplify the equation

$$= \sum_{k=1}^{n} \mathbf{k}_i^{(k)} \left(\mathbf{C}^{-1}\mathbf{k}_i\right)^{(k)}$$

Now, this can be easily extended to a single computation for all $i \in \mathcal{A}$ where $\mathcal{A}$ denotes the set of images in camera set A. $\mathbf{k}_i^{(k)} \left(\mathbf{C}^{-1}\mathbf{k}_i\right)^{(k)}$ can be seen as the k-th column of the element-wise multiplication of $\mathbf{K}_\mathcal{A}$ and $\mathbf{C}^{-1}\mathbf{K}_\mathcal{A}$ which results in

$$\mathbf{v}_\mathcal{A} = \begin{pmatrix} \vdots \\ \mathbf{k}_i^T\mathbf{C}^{-1}\mathbf{k}_i \\ \vdots \end{pmatrix}_{i\in\mathcal{A}} = \sum_{k=1}^{n} \left(\mathbf{K}_\mathcal{A} \odot \left(\mathbf{C}^{-1}\mathbf{K}_\mathcal{A}\right)\right)_k$$

summing up all the columns of $(\mathbf{K}_\mathcal{A} \odot \left(\mathbf{C}^{-1}\mathbf{K}_\mathcal{A}\right)$. Similarly for camera set B the term $\mathbf{k}_j^T\mathbf{C}^{-1}\mathbf{k}_j$ for all $j \in \mathcal{B}$ can be computed as

$$\mathbf{v}_\mathcal{B} = \begin{pmatrix} \vdots \\ \mathbf{k}_j^T\mathbf{C}^{-1}\mathbf{k}_j \\ \vdots \end{pmatrix}_{j\in\mathcal{B}} = \sum_{k=1}^{n} \left(\mathbf{K}_\mathcal{B} \odot \left(\mathbf{C}^{-1}\mathbf{K}_\mathcal{B}\right)\right)_k$$

For getting a single sum of matrices to compute the distance for all pairs $(i, j), i \in \mathcal{A}$, $j \in \mathcal{B}$ both vectors have to be repeated so that for the same image i the first term remains the same and for the same image j the second term remains the same:

$$\mathbf{V}_{\mathcal{A}} = \begin{array}{c} \begin{array}{ccc} 1 & \dots & |\mathcal{B}| \end{array} \\ \begin{pmatrix} \mathbf{v}_{\mathcal{A}} & \dots & \mathbf{v}_{\mathcal{A}} \end{pmatrix} \end{array}, \qquad \mathbf{V}_{\mathcal{B}} = \begin{array}{c} 1 \\ \vdots \\ |\mathcal{A}| \end{array} \begin{pmatrix} \mathbf{v}_{\mathcal{B}}^T \\ \vdots \\ \mathbf{v}_{\mathcal{B}}^T \end{pmatrix}$$

Finally, the pairwise Mahalanobis distance can be computed for all combinations at once resulting in this distance matrix

$$\mathbf{D}_{\mathcal{AB}} = \mathbf{V}_{\mathcal{A}} + \mathbf{V}_{\mathcal{B}} - 2\mathbf{K}_{\mathcal{A}}^T \mathbf{C}^{-1} \mathbf{K}_{\mathcal{B}}$$

The last steps of our implementation only make sure to return for every image from camera set A the r (rank) best images from camera set B, meaning the ones with the lowest distance. The performance (matching rate) of the model is computed as the number of cases (images in camera set A) a matching image is among the r images with lowest distance divided by the total number of cases. Further a visualization is included, showing the r images with the lowest distance for specified images from camera set A. Figure 7 shows an example of this visualization.

4 Tests

Validation and testing are tools that are unavoidable in order to make a machine learning algorithm generalize and perform well on unseen data. Our aim was to get rank values similar to those reported by Nguyen and De Baets [6] and Köstinger et al. [5].

For this purpose, we experimented with different Kernel methods implemented in the `kernlab` package:

- `rbfdot` Radial Basis kernel function
- `polydot` Polynomial kernel function
- `vanilladot` Linear kernel function

The linear kernel does not have any further hyperparameter to tune. For the RBF kernel we tested σ within of a range of $[10^{-4}; 10^{3}]$ with logarithmic increments. In the `kernlab` package the RBF kernel method is implemented as $k(x, x') = \exp\left(-\sigma \|x - x'\|^2\right)$. Furthermore, for the polynomial kernel we chose degrees $\in [2; 10]$. Moreover, we changed the regularization parameter ϵ. For this, we used a logarithmic sequence of values in the range of $[10^{-10}; 10^{-1}]$.

We applied the grid analysis technique to optimize the hyperparameter combinations. Before applying the technique it is important to divide the dataset of

rank	data - matching rate				
	iLIDS	CAVIAR4REID	3DPeS	PRID450S	CUHK01
1	44.0	41.9	48.7	53.9	54.3
5	70.05	71.5	72.6	81.0	74.2
10	82.6	85.5	83.9	88.8	80.5
20	93.4	96.1	92.1	94.5	87.0

(a) k-KISSME [6]

Rank	Matching Rate
1	19.6
10	62.2
25	80.7
50	91.8

(b) KISSME [5] on the VIPeR dataset [3]

Figure 6: Matching rates of the k-KISSME algorithm (a) and the ordinary KISSME (b) on different datasets

the images into training, validation and test sets as described in Section 3. Afterwards, we can compute the constraints and covariance matrix on the training data. We optimize the metric's hyperparameters over the sum of rank matching rates of the validation set. To compare our results with the ones in the papers we also used ranks $1, 5, 10, 20, 25, 50$. The final matching rate of a rank is the percentage to which within the first $x \in 1, 5, 10, 20$ images the person p is re-identified correctly.

Grid Search Grid Search is the most basic hyperparameter tuning method. With this technique we simple apply our metric for each possible combination of all the hyper-parameter values provided. Afterwards evaluating each metric call and selecting the one which produces the best results.

Figure 6 shows how the implementations from the paper perform on different datasets. From Fig. 6, Table 6a we can see, that it is possible to achieve overall good matching rates with the k-KISSME method. The matching rates of the KISSME algorithm on the VIPeR dataset (Table 6b) are significantly worse but this could also be dependent on the data.

Without hyperparameter tuning, using $\epsilon = 0.001$ and the linear kernel function we achieve matching rates as shown in Table 1a. With our described hyperparameter optimization we obtain the following best-performing combination: $\epsilon = \mathbf{1 * 10^{-6}}$ and **linear** kernel method.

Applying this configuration on the test dataset we achieve the performance shown in Table 1b. Thus, we were able to slightly improve the performance of our k-KISSME method. Figure 7 shows a visual representation of rank 5 matching results for four randomly selected query images. It can be seen that for the first, third and fourth query the algorithm was able to find the correct match with rank 1 whereas for the second row, no correct match was found. Comparing our results with those of the KISSME method (Table 6b) we can see that our method outperforms their approach by 5% for rank 1 and 25. On the other hand, for rank 10 and 50 our implementation performs marginally worse. Furthermore, we did not reach

Rank	Matching Rate
1	23.81
5	42.86
10	57.14
20	76.19
25	77.28
50	81.03

(a) without

Rank	linear kernel
1	26.98
5	49.21
10	60.32
20	84.13
25	85.71
50	90.48

(b) with

Table 1: Our results without (a) and with (b) hyperparameter optimization on the validation set

the results of the kernelized KISSME implementation by Nguyen and De Baets [6]. However, they did not test it on the VIPeR dataset which could be the reason for this. Further reasons are discussed in section 5.

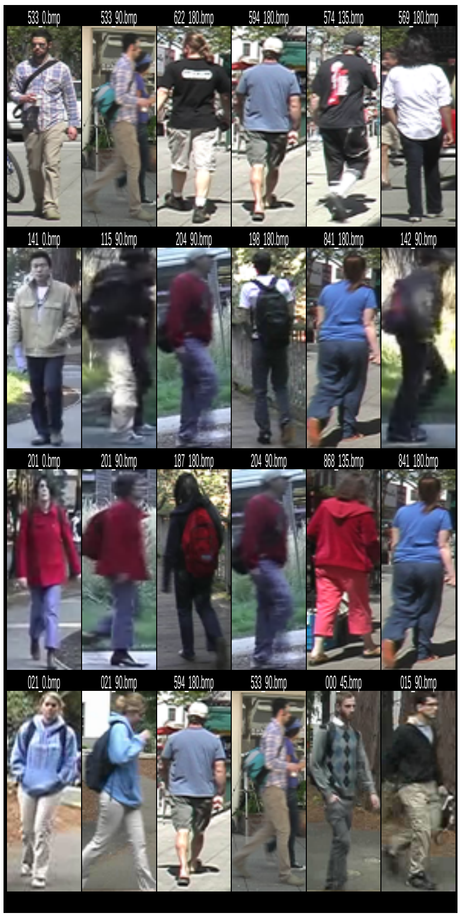

Figure 7: Visualization of the images with highest matching rate. The first column shows the query image, the other five columns the respective rank 5 matching results

5 Discussion

Our implementation is strongly geared to the paper by Nguyen and De Baets [6] and its related references. However, our results are only slightly comparable because we used a different dataset than those examined in the paper. Köstinger et al. [5] though used the VIPeR dataset in their KISSME paper but applied a different feature extraction. To get a better evaluation methodology, cross validation would achieve results with less variance and less influenced by coincidence and could also slightly improve the results. A cumulative matching characteristic (CMC) curve, which is analogous to the ROC curve would allow us to capture the performance over all ranks and not only the ranks we have chosen. Nevertheless, our tests give a good estimate of the matching rates and definitely perform comparable or even better than KISSME.
As already mentioned, compared to the results of k-KISSME our average scores are much worse for all ranks. Discrepancy can be explained by the difference in the datasets, which should be the main reason for the varying results. Moreover, the amount of must- and cannot links might also affect the results. We could generate more cannot-links to train with a bigger training set. We discarded this approach because of the computational costs and we did not want to introduce an imbalance of must- and cannot-links. Furthermore, we did not increase the sampling rate of our hyperparameter search because of the chance of overfitting on the validation data. To summarize, we explain the differences by:

- different training set
- not fully tuned hyperparameters
- different parameters in the feature extraction
- amount of must- and cannot links

6 Future work

Besides investigating the issues that we discussed in Section 5, for the future we suggest applying the proposed kernelized metric learning to further domains in order to verify its capabilities. First, we think of employing the algorithm for re-identification tasks on more visual domains such as facial recognition or object matching (e.g. for finding an item in an online shop using an image in which the item is embedded into some scenery). Köstinger et al. [5] already examined the original KISSME algorithm for faces and in the domain of matching toy cars. Their results can be used for comparison.

Additionally, Nguyen and De Baets [6] extended their approach to an incremental setting that addresses streams of data; only one pairwise constraint is added at a time. The approach is faster as it avoids expensive recomputations while proving to perform competitively to the non-incremental version. In the future, this approach should also be reviewed more extensively as it allows an online learning and constant improvement of the metric. Furthermore, different methods for the feature extraction should be analyzed for instance the enhanced Local Maximal Occurrence (eLOMO), an extension of the in Section 2.1 presented method, as proposed by Dong et al. [13]. The authors state that it can "resemble the coarse-to-fine recognition mechanism of human vision system, thus [...] provide a more discriminative descriptor for re-identification". [13]

Lastly, the advantage of the kernelization, as any kernel function can be applied to it, should be utilized to try learning a metric for non-vectorial data such as graph-like structures (networks, sequences, trees) and strings.

7 Conclusion

To sum up kernelized KISSME is an advanced method for person re-identification. It performs especially well for the challenging problem of comparing data from different cameras where one person can be shown under different environmental conditions and angles. We handled the issues related to illumination and viewpoint variance by applying the LOMO feature extraction that is capable of obtaining a robust representation of the images.

The scope of our project implementation includes adapting an existing LOMO feature extractor in Python to our needs and coding the k-KISSME algorithm in R. The kernelization of KISSME is an example how kernels can reduce computational complexity for datasets where only few instances are available, cases where methods such as deep learning or convolutional neural networks often fail. Employing our implementation to the VIPeR dataset shows that it is possible to re-identify people in around 27% of the test cases by only considering the best match. Furthermore, it reaches values of over 84% for rank 20 matches on our test data. As seen in the Discussion, the results are not directly comparable with the ones of [5] and [6]. This comes mainly due to the investigated dataset and the used feature extraction. In Section 6 potential additional fields of application and further steps are specified. The issues need to be examined to make this method more practical although even applying our method could mean a significant improvement compared to matching all images manually.
Considering the improvement in computational complexity especially for small datasets with plenty of features, further research on kernel methods like this is definitely advisable.

References

[1] E. Ahmed, M. Jones, and T. K. Marks. An improved deep learning architecture ofr person re-identification. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3908–3916, June 2015.

[2] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1249–1258, June 2016.

[3] Doug Gray, Shane Brennan, and Hai Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007.

[4] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 34–39. IEEE, 2014.

[5] Martin Köstinger, Martin Hirzer, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Large Scale Metric Learning from Equivalence Constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] Bac Nguyen and Bernard De Baets. Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Transactions on Image Processing*, 28(2):589–600, feb 2019.

[7] Shengcai Liao, Yang Hu, Xiangyu Zhu, and Stan Z. Li. Person re-identification by local maximal occurrence representation and metric learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2197–2206, June 2015.

[8] Edwin H. Land and John J. McCann. Lightness and retinex theory. *J. Opt. Soc. Am.*, 61(1):1–11, Jan 1971.

[9] D. J. Jobson, Z. Rahman, and G. A. Woodell. Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6(3): 451–462, March 1997.

[10] D. J. Jobson, Z. Rahman, and G. A. Woodell. A multiscale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976, July 1997.

[11] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikäinen, and S. Z. Li. Modeling pixel process with scale invariant local patterns for background subtraction in

complex scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1301–1306, June 2010.

[12] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[13] Husheng Dong, Ping Lu, Shan Zhong, Chunping Liu, Yi Ji, and Shengrong Gong. Person re-identification by enhanced local maximal occurrence representation and generalized similarity metric learning. *Neurocomputing*, 307:25–37, 2018.