Towards Visual Feature Translation

Jie Hu[1], Rongrong Ji[12]*, Hong Liu[1], Shengchuan Zhang[1], Cheng Deng[3], and Qi Tian[4]
[1]Fujian Key Laboratory of Sensing and Computing for Smart City, Department of Cognitive Science, School of Information Science and Engineering, Xiamen University, Xiamen, China.
[2]Peng Cheng Laboratory, Shenzhen, China. [3]Xidian University. [4]Noah's Ark Lab, Huawei.
{hujie.cpp,lynnliu.xmu,chdeng.xd}@gmail.com, {rrji,zsc_2016}@xmu.edu.cn, tian.qi1@huawei.com

Abstract

Most existing visual search systems are deployed based upon fixed kinds of visual features, which prohibits the feature reusing across different systems or when upgrading systems with a new type of feature. Such a setting is obviously inflexible and time/memory consuming, which is indeed mendable if visual features can be "translated" across systems. In this paper, we make the first attempt towards visual feature translation to break through the barrier of using features across different visual search systems. To this end, we propose a Hybrid Auto-Encoder (HAE) to translate visual features, which learns a mapping by minimizing the translation and reconstruction errors. Based upon HAE, an Undirected Affinity Measurement (UAM) is further designed to quantify the affinity among different types of visual features. Extensive experiments have been conducted on several public datasets with sixteen different types of widely-used features in visual search systems. Quantitative results show the encouraging possibilities of feature translation. For the first time, the affinity among widely-used features like SIFT and DELF is reported.

1. Introduction

Visual features serve as the basis for most existing visual search systems. In a typical setting, a visual search system can only handle pre-defined features extracted from the image set offline. Such a setting prohibits the reusing of a certain kind of visual feature across different systems. Moreover, when upgrading a visual search system, a time-consuming step is needed to extract new features and to build the corresponding indexing, while the previous features and indexing are simply discarded. Breaking through such a setting, if possible, is by any means very beneficial. For instance, the existing features and indexing can be efficiently reused when updating old features with new ones, which can significantly save the time and memory cost. For another instance, images can be efficiently archived with only respective features for cross-system retrieval. These examples are detailedly depicted in Fig. 1.

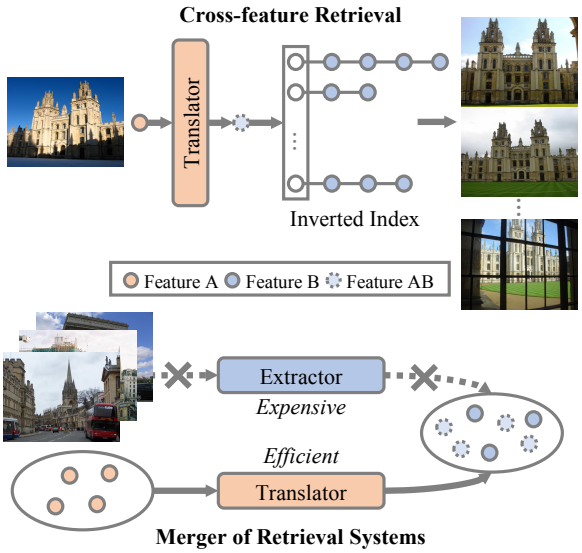

Figure 1. Two potential applications of visual feature translation. Top: In cross-feature retrieval, Feature A is translated to Feature AB, which can be used to search images that are represented and indexed by Feature B. Bottom: In the merger of retrieval systems, Feature A used in System A is efficiently translated to Feature AB, instead of the expensive process of re-extracting entire dataset in System A with Feature B.

However, feature reusing is not an easy task. Various dimensions and diverse distributions of different types of features prohibit reusing features directly. Therefore, a feature "translator" is needed to transform across different types of features, which, to our best knowledge, remains untouched in the literature. Intuitively, given a set of images extracted with different types of features, one can leverage the feature pairs to learn the corresponding feature translator.

In this paper, we make the first attempt to investigate *vi-*

*Corresponding author.

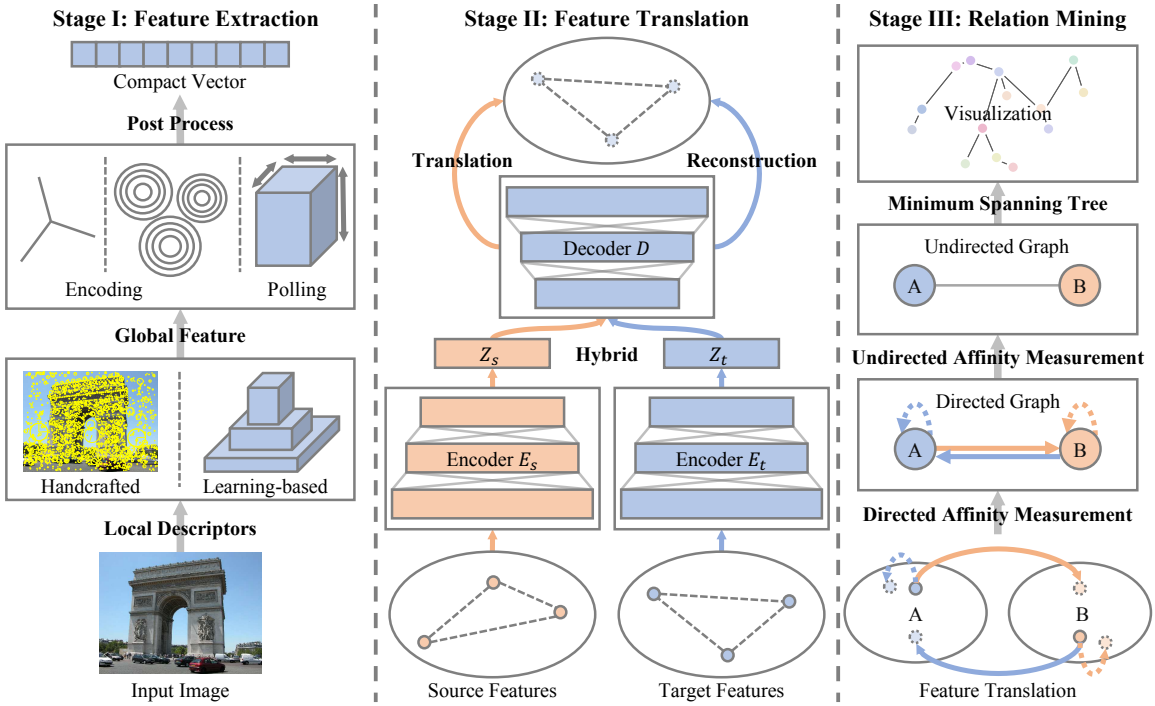

Figure 2. The overall flowchart of the proposed visual feature translation. In Stage I, different handcrafted or learning-based features are extracted from image set for training. In Stage II, the mappings from source features to target features are learned by our HAE with the encoders E_s, E_t and the decoder D. Then the encoder E_s and the decoder D are used in inference. In Stage III, the UAM is calculated to quantify the affinity among different types of visual features, which is further visualized by employing the Minimum Spanning Tree.

sual feature translation. Concretely, we propose a Hybrid Auto-Encoder (HAE) that learns a mapping from source features to target features by minimizing the translation and reconstruction errors. HAE consists of two encoders and one decoder. In training, the source and target features are encoded into a latent space by corresponding encoders. Features in this latent space are sent to a shared decoder to produce the translated features and reconstructed features. Then the reconstruction and translation errors are minimized by optimizing the objective function. In inference, the encoder of source features and the shared decoder are used for translation. The proposed HAE further provides a way to characterize the affinity among different types of visual features. Based upon HAE, an Undirected Affinity Measurement (UAM) is further proposed, which provides, also for the first time, a quantification of the affinity among different types of visual features. We also discover that UAM can predict the translation quality before the actual translation happens.

We train HAE on the Google-Landmarks dataset [16] and evaluate in total 16 different types of widely-used features in visual search community [2, 4, 19, 21, 29, 36, 41, 44, 52]. The tests of feature translation are conducted on three benchmark datasets, *i.e.*, Oxford5k [40], Paris6k [37], and Holidays [18]. Quantitative results show the encouraging possibility for feature translation. In particular, HAE works relatively well for feature pairs such as V-CroW to V-SPoC (*e.g.*, 0.1 mAP decrease on the Oxford5k benchmark) and R-rMAC to R-CroW (*e.g.*, 1.8 mAP decrease on the Holidays benchmark). Interestingly, visual feature translation provides some intriguing results (see Fig. 4 later in our experiments). For example, when translating from SIFT to DELF, characteristics like rotation or viewpoint invariance can be highlighted, which provides a new way to absorb merits of handcrafted features to learning-based ones.

In short, our contributions can be summarized as below:

- We are the first to address the problem of visual feature translation, which fills in the gaps between different types of features.
- We are the first to quantify the affinity among different types of visual features in retrieval, which can be used to predict the quality of feature translation.
- The proposed scheme innovates in several detailed designs, such as the HAE for training the translator and the UAM for quantifying the affinity. The source code and meta-data are released online[1].

The rest of this paper is organized as follows. Section 2 reviews the related work. The proposed feature translation and feature relation mining algorithms are introduced in Section 3. Quantitative experiments are given in Section 4. Finally, we conclude this work in Section 5.

[1]https://github.com/hujiecpp/VisualFeatureTranslation

2. Related Work

Visual Feature. Early endeavors mainly include holistic features (*e.g.*, color histogram [15] and shape [7]) and handcrafted local descriptors [6, 20, 30, 31, 33, 39, 47, 49], such as SIFT [29] and ORB [45]. Then, different aggregation schemes (*e.g.*, Fisher Vector [36] and VLAD [19]) are proposed to encode local descriptors. Along with the proliferation of neural networks, deep visual features have dominated visual search [1, 4, 5, 12, 16, 21, 32, 41, 43, 52], for instance, the local feature DELF [16] and the global feature produced by GeM [41] pooling are both prominent for representing images. Detailed surveys of visual features can be found in [50, 55].

Transfer Learning. Transfer learning [35, 51] aims to improve the learning of the target task using the knowledge in source domain. It can be subdivided into: instance transfer, feature transfer, parameter transfer, and relation transfer. Our work relates to, but is not identical with, the feature transfer. Feature transfer [3, 9, 11, 13, 24, 27, 34, 42, 53] is usually based on the hypothesis that the source domain and target domain have some shared characteristics. It aims to find a common feature space for both source and target domains, which serves as a new representation to *improve the learning of the target task*. For instance, the Structural Corresponding Learning [8] uses pivot features to learn a mapping from features of both domains to a shared feature space. For another instance, Joint Geometrical and Statistical Alignment [54] learns two coupled projections that project features of both domains into subspaces where the geometrical and distribution shifts are reduced. More recently, deep learning has been introduced into feature transfer [25, 26, 28, 46], in which neural networks are used to find the common feature spaces. In contrast, the visual feature translation aims to learn a mapping to translate features from the source space to the target space, and the translated features are *used directly in the target space*.

3. Visual Feature Translation

Fig. 2 shows the overall flowchart of the proposed visual feature translation. Firstly, source and target feature pairs are extracted from image set for training in Stage I. Then, feature translation based on HAE is learned in Stage II. After translation, the affinity among different types of features is quantified and visualized in Stage III.

3.1. Preprocessing

As shown in Stage I of Fig. 2, we prepare the source and target features for training the subsequent translator. For the handcrafted features such as SIFT [29], the local descriptors are extracted by the designed procedures firstly. These local descriptors are then aggregated by encoding schemes to produce the global features. For the learning-based features such as V-MAC [44, 52], the feature maps are extracted by neural networks firstly, followed by a pooling layer or encoding schemes to produce the feature vectors. In our settings, we investigate in total 16 different types of features, a detailed table of which can be found in Table 1. The feature sets are arranged to form 16×16 feature set pairs $(\mathcal{V}_s, \mathcal{V}_t)$, where $\mathcal{V}_s$ denotes the set of source features and $\mathcal{V}_t$ denotes the set of target features. The implementation is detailed in Section 4.1.

Algorithm 1 The Training of HAE

Input: Feature sets $\mathcal{V}_s$ and $\mathcal{V}_t$, decoders E_s, E_t and encoder D parameterized by $\theta_{E_s}, \theta_{E_t}$ and θ_D.
Output: The learned translator E_s and D.

```
while not convergence do
    Get Z_s by Z_s = E_s(V_s).
    Get Z_t by Z_t = E_t(V_t).
    Get V_st by translation: V_st = D(Z_s).
    Get V_tt by reconstruction: V_tt = D(Z_t).
    Optimize the Eq. 1.
end while
return E_s and D.
```

3.2. Learning to Translate

To achieve the task of translating different types of features, a Hybrid Auto-Encoder (HAE) is proposed, which is shown in Stage II of Fig. 2. For training HAE, the source features $\mathcal{V}_s$ and the target features $\mathcal{V}_t$ are input to the model which outputs the translated features $\mathcal{V}_{st}$ and the reconstructed features $\mathcal{V}_{tt}$.

Formally, HAE consists of two encoders E_s, E_t and one decoder D. In training, $v_s \in \mathcal{V}_s$ is encoded into the latent feature $z_s \in \mathcal{Z}_s$ by the encoder E_s, and the same for $v_t \in \mathcal{V}_t$ into $z_t \in \mathcal{Z}_t$ by E_t. The latent features z_s and z_t are then decoded to obtain the translated feature $v_{st} \in \mathcal{V}_{st}$ and the reconstructed feature $v_{tt} \in \mathcal{V}_{tt}$ by the shared decoder D. We define the Euclidean distance as $\mathcal{E}(x, y) = \|x - y\|_2$. The E_s, E_t and D are parameterized by $\theta_{E_s}, \theta_{E_t}$ and θ_D, which is learned by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}(\theta_{E_s}, \theta_{E_t}, \theta_D) =& \mathbb{E}_{v_{st} \in \mathcal{V}_{st}, v_t \in \mathcal{V}_t}[\mathcal{E}(v_{st}, v_t)] \\ &+ \mathbb{E}_{v_{tt} \in \mathcal{V}_{tt}, v_t \in \mathcal{V}_t}[\mathcal{E}(v_{tt}, v_t)],\end{aligned} \tag{1}$$

where we define the first item as the translation error and the second item as the reconstruction error.

In the processing of the feature translation, only E_s and D are used to translate features from $\mathcal{V}_s$ to $\mathcal{V}_t$. The algorithm for training the HAE is summarized as Alg. 1.

We then get the following characteristics for our visual feature translation:

Characteristic I: Saturation. The performance of translated features is difficult to exceed that of the target features.

This phenomenon is inherent in the feature translation process. According to Eq. 1, the translation and reconstruction errors are minimized after optimizing. However, they are difficult to approach zero due to the information loss brought by the architecture of Auto-Encoder.

Characteristic II: Asymmetry. The convertibility of translation is discrepancy between A2B and B2A (We abbreviate A2B for the translation from features A to features B, *etc*.). The networks for translating different types of features are by nature asymmetry. HAE relies on the translation error and reconstruction error, which is not the same between A2B and B2A.

Characteristic III: Homology. In general, homologous features tend to have high convertibility. In contrast, the convertibility is not guaranteed for heterogenous features. Homologous features refer to the features extracted by the same extractor but encoded or pooled by different methods (*e.g*., DELF-FV [16, 36] and DELF-VLAD [16, 19], or V-CroW [21] and V-SPoC [4]), and the heterogenous features refer to the features extracted by different extractor. This characteristic is analyzed in details in Section 4.2.

3.3. Feature Relation Mining

HAE provides a way to quantify the affinity between feature pairs. Therefore, the affinity among different types of features can be quantified as the Stage III shown in Fig. 2. First, we use the difference between translation and reconstruction errors as a Directed Affinity Measurement (DAM) and calculate the directed affinity matrix M which forms a directed graph for all feature pairs. Second, in order to quantify the total affinity among features, we design an Undirected Affinity Measurement (UAM) by employing M. The calculated undirected affinity matrix U is symmetry, which forms a complete graph. Third, we visualize the local similarity between features by using the Minimum Spanning Tree (MST) of the complete graph.

Directed Affinity Measurement. We assume that after optimizing, for Eq. 1, the reconstruction error is smaller than the translation error. This intuitive assumption is verified later in Section 4.3. Then, we can find that:

$$\begin{aligned}\mathcal{L} \geq & \mathbb{E}_{v_{st}\in\mathcal{V}_{st},v_t\in\mathcal{V}_t}[\mathcal{E}(v_{st},v_t)] \\ & -\mathbb{E}_{v_{tt}\in\mathcal{V}_{tt},v_t\in\mathcal{V}_t}[\mathcal{E}(v_{tt},v_t)] \geq 0.\end{aligned} \tag{2}$$

According to this inequation, when minimizing $\mathcal{L}$, the translation error is forced to approximate the reconstruction error. If translation error is close to reconstruction error, we think the translation between source and target features is similar to the reconstruction of target features, which indicates the source and target features have high affinity. Therefore, we regard the difference between the translation and reconstruction errors as the affinity measurement. We use $M_{s\to t}$ to represent the DAM between $\mathcal{V}_s$ and $\mathcal{V}_t$. The calculation of the element at row s and column t of M is defined as follows:

$$\begin{aligned}M_{s\to t} = & \mathbb{E}_{v_{st}\in\mathcal{V}_{st},v_t\in\mathcal{V}_t}[\mathcal{E}(v_{st},v_t)] \\ & -\mathbb{E}_{v_{tt}\in\mathcal{V}_{tt},v_t\in\mathcal{V}_t}[\mathcal{E}(v_{tt},v_t)].\end{aligned} \tag{3}$$

Algorithm 2 Affinity Calculation and Visualization

Input: The number of different types of features n, the feature pairs $(\mathcal{V}_s, \mathcal{V}_t)$ and the translator E_s, D.

Output: The directed affinity matrix M and the undirected affinity matrix U.

```
for i = 1 : n, j = 1 : n do
    Calculate M_{i→j} by Eq. 3.
end for
for i = 1 : n, j = 1 : n do
    Calculate R_{i→j} and C_{i→j} by Eq. 4 and Eq. 5.
end for
Calculate U by Eq. 6.
Generate the MST based on U by Kruskal's algorithm.
Visualize the MST.
return M, U.
```

Undirected Affinity Measurement. Due to the asymmetry characteristic, M is asymmetric, which is unsuitable to be the total affinity measurement of feature pairs. We then resort to designing an Undirected Affinity Measurement (UAM) to quantify the overall affinity among different types of features. Specifically, we treat A2B and B2A as a unified whole, therefore the rows and columns of M are considered consistently. For the rows of M, the element at row i and column j of the matrix R with normalized rows is defined as:

$$R_{i\to j} = \frac{M_{i\to j} - \min(M_{i\to :})}{\max(M_{i\to :}) - \min(M_{i\to :})}, \tag{4}$$

where $\min(M_{i\to :})$ and $\max(M_{i\to :})$ are the minimum and maximum of the row i, and $R_{i\to j}$ is normalized to $[0, 1]$.

In a similar way, for the columns of M, the element at row i and column j of the matrix C with normalized columns is defined as:

$$C_{i\to j} = \frac{M_{i\to j} - \min(M_{:\to j})}{\max(M_{:\to j}) - \min(M_{:\to j})}, \tag{5}$$

where $\min(M_{:\to j})$ and $\max(M_{:\to j})$ are the minimum and maximum of the column j, and $C_{i\to j}$ is normalized to $[0, 1]$.

The undirected affinity matrix U is defined as follows:

$$U = \frac{1}{4}(R + R^T + C + C^T). \tag{6}$$

If U_{ij} has a small value, feature i and feature j are similar, and vice versa.

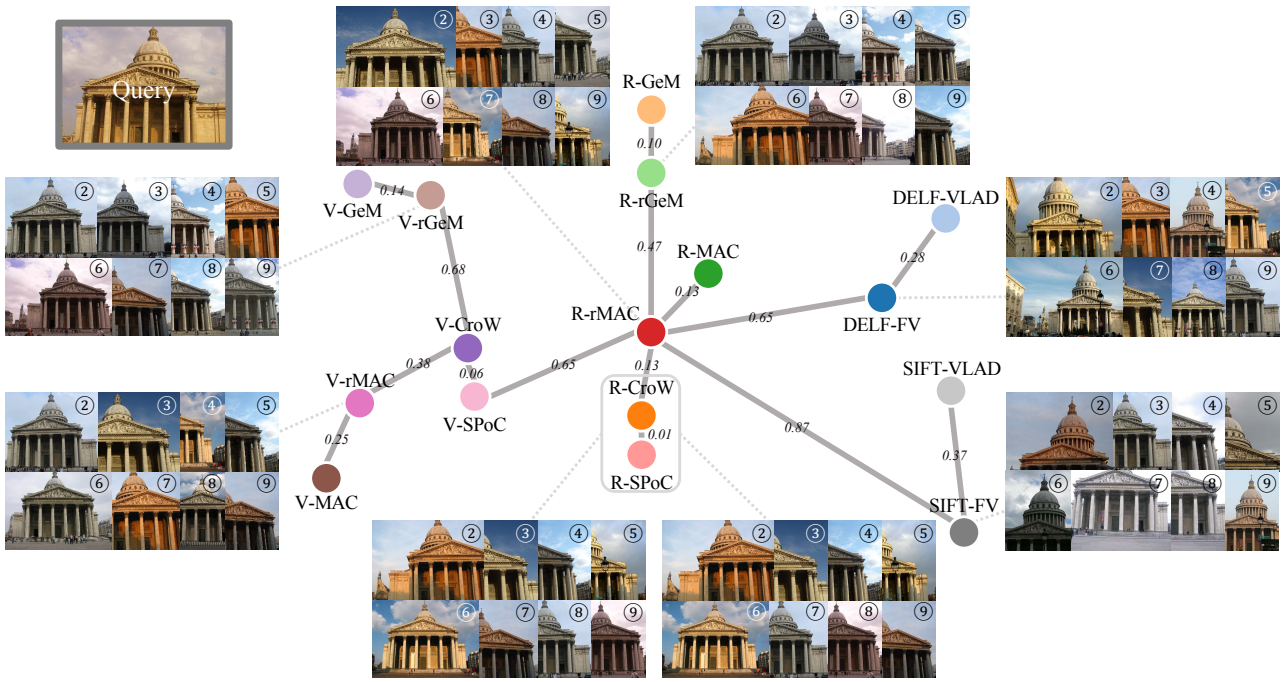

Figure 3. The visualization of the MST based on U with popular visual search features. The length of edges is the average value of the results on Holidays, Oxford5k and Paris6k datasets. The images are the retrieval results for a query image of the Pantheon with corresponding features in the main trunk of the MST. The close feature pairs such as R-SPoC and R-CroW have similar ranking lists.

The Visualization. We use the Minimum Spanning Tree (MST) to visualize the relationship of features based on U. The Kruskal's algorithm [23] is used to find MST. This algorithm firstly creates a forest G, where each vertex is a separate tree. Then the edge with minimum weight that connects two different trees is recurrently added to the forest G, which combines two trees into a single tree. The final output forms an MST for the complete graph. The MST helps us to understand the most related feature pairs (connected by an edge), as well as their affinity score (the length of the edge). The overall procedure is summarized as Alg. 2. The visualization result of the affinity among popular visual features with a query example can be found in Fig. 3.

4. Experiments

We show the experiments in this section. First, we introduce the experimental settings. Then, the translation performance of our HAE is reported. Finally, we visualize and analyze the results of relation mining.

4.1. Experimental Settings

Training Dataset. The Google-Landmarks dataset [16] contains more than 1M images captured at various landmarks all over the world. We randomly pick 40,000 images from this dataset to train HAE, and pick 4,000 other images to train PCA whitening [4, 17] and creating the codebooks for local descriptors.

Test Dataset. We use the Holidays, Oxford5k and Paris6k datasets for testing. The Holidays dataset [18] has 1,491 images with various scene types and 500 query images. The Oxford5k dataset [37] consists of 5,062 images which have been manually annotated to generate a comprehensive ground truth for 55 query images. Similarly, the Paris6k dataset [38] consists of 6,412 images with 55 query images. Since the scalability of retrieval algorithms is not our main concern, we do not use the disturbance dataset Flickr100k [38]. Recently, the work in [40] revisited the labels and queries on both Oxford5k and Paris6k. Because the images remained the same, which does not affect the characteristics of features, we do not use the revisited datasets as our test datasets. The mean average precision (mAP) is used to evaluate the retrieval performance. We translate the source features of reference images to the target space, and the target features of query images are used for testing.

Features. L1 normalization and square root [2] are applied to SIFT [29]. The original extraction approach (at most 1,000 local representations per image) is applied to DELF [16]. The codebooks of FV [36] and VLAD [19] are created for SIFT and DELF. We use 32 components of Gaussian Mixture Model (GMM) to form the codebooks of FV and the dimension of this feature is reduced to 2,048 by PCA whitening. The aggregated features are termed as SIFT-FV and DELF-FV. We use 64 central points to form the codebooks of VLAD and the dimension

	Holidays	Oxford5k	Paris6k
DELF-FV [16, 36]	83.42	73.38	83.06
DELF-VLAD [16, 19]	84.61	75.31	82.54
R-CroW [21]	86.38	61.73	75.46
R-GeM [41]	89.08	84.47	91.87
R-MAC [44, 52]	88.53	60.82	77.74
R-rGeM [41]	89.32	84.60	91.90
R-rMAC [52]	89.08	68.46	83.00
R-SPoC [4]	86.57	62.36	76.75
V-CroW [21]	83.17	68.38	79.79
V-GeM [41]	84.57	82.71	86.85
V-MAC [44, 52]	74.18	60.97	72.65
V-rGeM [41]	85.06	82.30	87.33
V-rMAC [52]	83.50	70.84	83.54
V-SPoC [4]	83.38	66.43	78.47
SIFT-FV [2, 29, 36]	61.77	36.25	36.91
SIFT-VLAD [2, 29, 19]	63.92	40.49	41.49

Table 1. The mAP (%) of target features.

of this feature is also reduced to 2,048 by PCA whitening. The aggregated features are termed as SIFT-VLAD and DELF-VLAD. For off-the-shelf deep features, we use ImageNet [10] pre-trained VGG-16 (abbreviated as V) [48] and ResNet101 (abbreviated as R) [14] to produce the feature maps. The max-pooling (MAC) [44, 52], average-pooling (SPoC) [4], weighted sum-pooling (CroW) [21], and regional max-pooling (rMAC) [52] are then used to pool the feature maps. The extracted features are termed as V-MAC, V-SPoC, V-CroW, V-rMAC, R-MAC, R-SPoC, R-CroW and R-rMAC, respectively. For fine-tuned deep features, we consider the generalized mean-pooling (GeM) and regional generalized mean-pooling (rGeM) [41]. The extracted features are termed as V-GeM, V-rGeM, R-GeM and R-rGeM, respectively.

Network Architecture. The task-specific network architectures of HAE have a fixed latent feature space of 510 dimension. The parameter settings of encoder which consists of fully-connect layers with ReLU-based activation function are 2048-2048-2048-510 or 512-512-510 for encoding the features with 2048 or 512 dimension. The parameter settings of the decoder are in reverse of that of encoder, depending on the dimension of the output features. The output features are L2 normalized. We use Multi-Layer Perceptron (MLP) as our baseline, whose architecture are 2048-2048-2048 or 512-512-512 for encoding the features with 2048 or 512 dimension, and the encoders are in reverse. We used Adam [22] optimizer to minimize the objective function for all feature pairs, where the learning rate is set as 0.00001.

4.2. Translation Results

Quantitative Evaluation. The performance of target features is shown in Table 1. We use the mAP difference between target and translated features to show the translation results. As shown in Table 2, we use a color map which is normalized according to the minimum (white) and maximum (colored) values to show results of each dataset. From the result, we find although there are still few differences between datasets, the trend of the colored values is almost the same.

For further analyzing, the results can be divided into three groups: high convertibility, inferior convertibility and low convertibility. Firstly, the high convertibility results appear mostly in the translation between homologous features. For example, when translating from V-CroW to V-SPoC, the mAPs drop 3.8, 0.1, 0.3 on the Holidays, Oxford5k and Paris6k datasets, respectively. Secondly, the inferior results are found between heterogenous features such as R-based features and V-based features. For example, when translating from R-GeM to V-GeM, the mAPs decrease 5.7, 11.3, 2.3 on the three datasets, respectively. Another example is the translation from V-rGeM to R-rMAC, the mAPs decrease 12.4, 7.1, 5.8 on the three datasets, respectively. Thirdly, the low convertibility results also emerge between heterogenous features. For example, when translating from SIFT-FV to DELF-FV, the performance is not high. Another example is the translation from DELF-VLAD to R-GeM, in which the former is extracted by Resnet50 and the latter is extracted by Resnet101. We explain it from the different depth of network architectures, different training procedures and different encoding/pooling schemes.

The average mAP difference of HAE compared with MLP on three datasets is shown in Table 3. From the results, we can see MLP has a very unstable performance. In contrast, HAE with appropriate dimension of latent feature performs better than MLP, due to the regularization effect brought by the "bottleneck" architecture (which enforces encoder to learn the most valuable information for decoder).

Qualitative Evaluation. Some cross-feature retrieval results are shown in Fig. 4. The first column shows a successful translation from V-CroW to V-SPoC, the ranking lists are almost the same. The second column shows an inferior translation from R-GeM to V-GeM. Interestingly, when querying an image of the *Arc de Triomphe* at night, the images of the *Arc de Triomphe* during the day are retrieved by the translated features and get high ranks, which inspires the integration of feature translation to improve cross-modal retrieval. The most exciting result lies in the third column: although the translation from SIFT-FV to DELF-FV suffers a low performance, the characteristics like rotation or viewpoint invariance can be highlighted by translation, which well bridges the merits of the handcrafted features to the learning-based features. For example, the images from the bottom view of the *Eiffel Tower* and the *Arc de Triomphe* get high ranks (both at Rank@4). The rotated images of them also have high ranks (at Rank@7 and Rank@3). Then, in the fourth column, we show these characteristics do not symmetrically exist in the reverse translation from DELF-

	DELF-FV	DELF-VLAD	R-CroW	R-GeM	R-MAC	R-rGeM	R-rMAC	R-SPoC	V-CroW	V-GeM	V-MAC	V-rGeM	V-rMAC	V-SPoC	SIFT-FV	SIFT-VLAD
DELF-FV	1.7	4.4	16.0	19.3	20.8	17.9	13.0	16.9	11.4	14.1	20.9	13.0	11.0	13.6	40.0	42.5
DELF-VLAD	4.0	3.0	15.9	19.1	21.3	17.7	14.0	16.1	10.5	14.0	21.0	11.2	11.3	12.2	40.7	42.9
R-CroW	12.9	16.2	1.2	8.6	4.9	7.4	3.2	2.6	8.8	13.1	17.9	10.5	8.2	8.5	32.6	38.4
R-GeM	10.2	13.4	6.4	1.8	5.5	2.1	3.0	5.0	6.7	5.7	12.6	5.3	7.1	8.8	33.5	38.1
R-MAC	12.7	15.1	2.8	8.1	4.1	7.7	1.8	2.7	6.2	12.3	8.8	10.3	4.6	7.3	38.5	41.7
R-rGeM	11.3	12.8	4.5	2.1	6.0	1.7	2.7	5.5	9.3	6.7	13.7	4.7	5.8	9.9	35.8	40.0
R-rMAC	11.6	14.8	1.8	8.6	4.3	8.0	2.0	3.3	7.6	10.6	11.6	8.9	5.2	9.5	37.2	40.6
R-SPoC	12.6	15.7	1.4	9.1	5.1	8.0	2.9	2.6	8.1	13.0	18.7	11.0	7.8	8.2	31.5	36.7
V-CroW	18.8	20.0	15.1	17.7	14.8	18.4	12.1	15.3	2.6	9.8	3.0	9.8	2.2	3.8	35.2	38.1
V-GeM	17.8	19.6	18.3	14.0	21.0	15.2	13.5	20.1	6.8	3.5	6.7	2.8	5.9	9.8	34.8	38.4
V-MAC	33.5	36.7	33.7	34.6	31.1	35.3	22.2	35.8	11.4	18.9	6.7	20.9	7.3	15.2	46.9	50.5
V-rGeM	18.0	19.9	17.2	15.0	20.2	12.7	12.4	17.5	8.9	2.4	9.9	1.4	5.8	10.4	35.4	37.4
V-rMAC	23.3	26.1	21.5	25.9	21.5	23.3	14.1	22.9	6.6	12.8	4.7	12.6	3.6	9.9	42.8	45.1
V-SPoC	17.2	18.0	13.6	16.8	14.7	16.5	11.1	13.4	1.8	10.3	5.7	8.1	3.6	2.2	30.9	36.6
SIFT-FV	55.9	63.8	61.2	68.5	69.5	66.6	57.1	59.3	59.5	60.8	63.4	60.3	59.4	54.9	3.7	4.9
SIFT-VLAD	57.9	63.6	61.4	69.7	70.7	67.3	56.0	60.9	60.5	59.7	64.8	60.4	60.4	55.9	1.6	5.9
DELF-FV	4.8	9.5	15.5	30.8	22.2	28.8	18.5	16.3	11.7	22.1	18.4	20.2	21.2	8.2	30.3	33.2
DELF-VLAD	5.2	4.2	10.4	27.0	11.8	25.5	13.7	9.7	8.4	19.6	22.8	17.4	17.9	7.9	26.8	30.1
R-CroW	27.2	27.2	2.1	24.4	5.2	21.3	8.3	2.8	16.8	27.1	21.1	21.3	20.3	15.3	27.9	30.6
R-GeM	19.3	15.8	1.5	2.6	0.9	3.1	5.1	3.4	12.8	11.3	18.6	11.2	14.9	12.5	31.6	33.9
R-MAC	30.5	28.0	8.0	26.4	5.8	25.8	9.8	6.9	17.6	27.9	20.6	26.8	23.9	18.3	30.6	33.3
R-rGeM	17.8	16.5	1.4	3.8	1.1	2.9	4.8	1.5	11.4	11.8	19.2	9.6	14.6	13.2	29.7	32.2
R-rMAC	26.5	24.6	0.9	21.0	1.3	19.1	4.9	1.5	15.3	22.4	17.1	18.9	16.4	15.3	28.2	31.8
R-SPoC	25.9	24.6	2.1	22.8	4.4	20.6	7.0	1.8	15.3	24.5	21.7	20.8	19.5	13.6	27.0	29.9
V-CroW	23.5	23.2	13.6	32.3	19.0	33.8	14.6	14.8	1.0	19.2	0.3	17.6	6.2	0.1	26.7	31.2
V-GeM	17.1	11.9	11.3	17.3	15.9	16.1	9.1	12.3	3.2	2.2	4.7	1.0	6.5	4.6	29.3	34.0
V-MAC	40.0	40.4	33.2	46.8	29.1	52.0	30.2	33.6	9.9	26.1	5.4	32.4	10.5	14.4	30.9	36.3
V-rGeM	18.1	13.4	9.7	21.6	16.0	17.0	7.1	10.8	3.8	4.1	6.8	1.9	6.3	7.1	25.8	29.7
V-rMAC	31.3	32.9	21.4	38.4	20.3	39.0	18.4	22.0	3.7	17.1	0.9	16.9	1.3	5.1	27.6	32.9
V-SPoC	24.7	22.5	14.8	38.3	17.9	36.4	17.0	16.0	2.1	19.1	3.5	17.3	6.6	0.5	24.6	30.6
SIFT-FV	65.3	67.9	55.4	80.9	56.2	79.5	61.0	57.2	61.0	77.1	56.4	75.3	64.1	59.2	9.5	13.5
SIFT-VLAD	63.4	67.3	57.0	81.2	56.5	79.7	61.4	57.2	59.9	76.6	56.5	75.2	63.2	57.6	10.2	9.8
DELF-FV	3.7	6.0	9.5	20.1	14.1	18.8	13.4	13.4	7.2	13.4	17.9	13.5	13.7	6.3	14.5	16.5
DELF-VLAD	6.4	3.3	8.9	18.0	13.0	16.0	11.5	10.7	5.5	12.3	19.1	12.9	13.7	6.5	15.0	19.0
R-CroW	16.8	17.3	4.0	17.4	6.0	15.1	8.9	4.9	11.0	14.4	18.4	13.5	14.9	11.1	17.6	22.7
R-GeM	10.7	8.3	0.2	3.5	1.9	2.9	0.6	1.1	2.7	2.3	5.7	4.0	7.6	3.9	19.4	20.5
R-MAC	18.9	18.8	4.7	15.9	7.9	14.2	9.0	6.5	12.3	14.2	18.0	13.8	18.1	12.2	20.6	25.2
R-rGeM	9.3	9.4	2.7	3.5	1.1	3.2	1.1	0.4	3.4	3.7	7.0	3.9	6.6	3.8	17.1	20.7
R-rMAC	14.2	13.6	0.2	13.7	3.4	10.4	4.9	0.9	6.9	9.6	11.9	8.9	9.8	7.1	18.2	22.0
R-SPoC	15.2	15.2	3.3	17.1	5.1	14.3	8.0	3.6	10.0	13.6	15.8	12.7	13.2	9.7	16.8	22.0
V-CroW	18.1	20.0	10.4	22.9	13.9	23.2	13.9	13.5	1.0	10.9	1.7	9.6	5.0	0.3	19.3	21.6
V-GeM	10.6	12.2	7.4	11.4	8.0	11.3	6.7	10.0	1.8	1.9	1.4	2.0	4.8	2.3	13.5	17.7
V-MAC	29.6	33.0	24.9	31.4	24.7	34.6	23.6	29.3	8.7	15.1	7.3	16.3	9.7	9.2	26.8	30.4
V-rGeM	10.9	12.8	6.2	12.3	6.8	12.3	5.8	6.5	1.2	3.3	3.1	1.2	4.8	1.7	12.6	16.0
V-rMAC	21.4	24.2	12.9	25.0	19.5	22.3	15.4	14.9	1.7	9.7	1.0	8.5	2.7	1.8	20.0	24.2
V-SPoC	16.9	21.8	11.6	23.8	14.2	25.6	14.1	13.8	2.5	13.2	2.7	12.7	6.5	1.6	15.4	19.7
SIFT-FV	59.8	64.0	59.3	82.2	67.3	78.9	62.3	61.7	63.0	71.6	63.0	68.9	66.9	60.1	8.8	10.0
SIFT-VLAD	58.7	60.7	60.7	80.5	67.4	78.2	63.5	61.4	62.3	70.9	63.6	68.7	65.8	59.1	5.9	10.2

Table 2. The mAP(%) difference between target and translated features on three public datasets: Holidays (Green), Oxford5k (Blue) and Paris6k (Brown) in the first, second and third blocks, respectively.

FV to SIFT-FV. We explain it from the limited representative ability of the SIFT-FV.

4.3. Relation Mining Results

After calculating the directed affinity matrix M and the undirected affinity matrix U, we average the values of the three datasets and draw the heat maps. As shown in Fig. 5 (left), the values of directed affinity matrix M verify our assumption that the reconstruction error is smaller than the translation error as all the values are positive. As shown in Fig. 5 (right), the positions of light and dark colors are almost the same as that of the translation results in Table 2, which indicates the UAM can be used to predict the translation quality between two given features. To study the relationship between features better, we visualize the MST based on U as Fig. 3. The images are the ranking lists for a query image with corresponding features. Since the results of leaf nodes connected in the MST (*e.g.* R-CroW and R-SPoC) are very similar, we mainly show the results of nodes in the trunk of the MST. The closer features return

(a) V-CroW to V-SPoC (b) R-GeM to V-GeM (c) SIFT-FV to DELF-FV (d) DELF-FV to SIFT-FV

Figure 4. The retrieval results for querying images of the *Eiffel Tower* (up) and the *Arc de Triomphe* (down) with the target features and the translated features. The images are resized for better view and the interesting results are colored by red bounding boxes.

	DELF-FV	DELF-VLAD	R-CroW	R-GeM	R-MAC	R-rGeM	R-rMAC	R-SPoC	V-CroW	V-GeM	V-MAC	V-rGeM	V-rMAC	V-SPoC	SIFT-FV	SIFT-VLAD
DELF-FV	1.5	5.9	72.0	85.8	73.6	86.0	77.9	72.7	9.8	17.4	18.5	15.3	16.1	9.7	43.1	46.9
DELF-VLAD	4.8	1.7	71.8	85.7	73.1	86.0	77.4	72.4	9.4	14.5	20.1	13.9	16.9	9.0	43.3	47.1
R-CroW	76.7	77.3	1.1	83.3	3.5	15.2	4.6	1.6	11.2	18.3	19.7	14.9	13.6	10.9	42.8	47.0
R-GeM	76.9	77.2	71.2	1.0	58.8	0.9	9.4	72.7	7.0	6.6	12.7	5.9	9.7	7.0	43.2	47.0
R-MAC	77.0	77.5	2.9	65.6	2.3	84.9	4.0	3.6	12.7	18.6	15.8	16.9	14.8	12.5	43.2	47.2
R-rGeM	77.7	76.8	4.0	1.2	69.4	0.9	3.2	22.9	7.5	7.3	15.3	5.4	9.2	7.7	43.2	46.8
R-rMAC	76.1	76.7	0.6	25.7	0.1	10.5	1.1	0.2	9.9	15.7	14.1	12.3	10.0	9.4	43.1	47.0
R-SPoC	76.8	77.3	0.7	82.0	3.0	15.2	4.2	1.4	10.3	17.3	18.8	14.1	12.3	9.7	42.9	46.9
V-CroW	21.3	25.1	14.8	24.7	17.1	27.8	15.5	15.0	0.9	11.1	0.1	10.0	3.2	0.3	43.1	46.5
V-GeM	16.6	23.2	18.7	15.3	27.7	15.2	12.6	22.0	1.8	1.2	0.2	0.7	3.4	2.8	43.1	46.2
V-MAC	44.6	66.4	53.1	71.8	51.5	69.9	37.0	62.7	8.7	16.4	3.5	17.9	6.9	10.6	43.3	46.9
V-rGeM	20.7	24.3	14.2	18.7	20.3	13.6	10.3	12.9	3.0	2.6	3.5	1.1	3.4	3.2	43.0	46.3
V-rMAC	29.3	42.1	22.9	34.1	24.3	32.7	17.8	22.3	2.8	11.9	1.3	10.2	1.5	3.6	42.2	46.1
V-SPoC	19.6	22.6	13.3	26.9	16.3	25.3	15.8	14.4	1.9	12.5	2.0	10.7	4.9	0.9	41.1	45.8
SIFT-FV	78.3	79.1	72.9	87.0	74.2	87.0	78.6	73.8	59.8	74.6	67.8	77.1	76.2	57.3	4.1	12.6
SIFT-VLAD	78.0	78.8	72.9	87.0	73.9	86.8	78.5	73.6	59.6	77.1	67.9	80.6	77.6	57.9	8.6	4.3
DELF-FV	3.4	6.6	13.7	23.4	19.0	21.8	15.0	15.5	10.1	16.5	19.1	15.6	15.3	9.4	28.2	30.7
DELF-VLAD	5.2	3.5	11.7	21.4	15.4	19.7	13.1	12.2	8.1	15.3	21.0	13.8	14.3	8.9	27.5	30.7
R-CroW	18.9	20.2	2.4	16.8	5.4	14.6	6.8	3.4	12.2	18.2	19.1	15.1	14.5	11.7	26.0	30.6
R-GeM	13.4	12.5	2.6	2.6	1.5	2.7	2.9	3.1	7.4	6.4	12.3	6.9	9.8	8.4	28.2	30.8
R-MAC	20.7	20.6	5.2	16.8	5.9	15.9	6.8	5.4	12.0	18.1	15.8	16.9	15.5	12.6	29.9	33.4
R-rGeM	12.8	12.9	1.1	3.2	2.0	2.6	2.9	2.2	8.0	7.4	13.3	6.1	9.0	9.0	27.6	31.0
R-rMAC	17.4	17.7	1.0	14.4	3.0	12.5	3.9	1.9	9.9	14.2	13.5	12.3	10.5	10.6	27.9	31.5
R-SPoC	17.9	18.5	2.3	16.3	4.9	14.3	6.0	2.7	11.1	17.1	18.7	14.9	13.5	10.5	25.1	29.5
V-CroW	20.2	21.1	13.0	24.3	15.9	25.1	13.5	14.5	1.6	13.3	1.7	12.3	4.5	1.4	27.1	30.3
V-GeM	15.2	14.6	12.3	14.2	15.0	14.2	9.8	14.2	3.9	2.6	4.3	1.9	5.7	5.6	25.9	30.0
V-MAC	34.3	36.7	30.6	37.6	28.3	40.6	25.3	32.9	10.0	20.0	6.5	23.2	9.2	12.9	34.9	39.1
V-rGeM	15.7	15.4	11.0	16.3	14.3	14.0	8.4	11.6	4.6	3.3	6.6	1.5	5.6	6.4	24.6	27.7
V-rMAC	25.4	27.7	18.6	29.7	20.4	28.2	15.9	20.0	4.0	13.2	2.2	12.7	2.5	5.6	30.2	34.1
V-SPoC	19.6	20.8	13.4	26.3	15.6	26.2	14.1	14.4	2.1	14.2	4.0	12.7	5.6	1.4	23.6	29.0
SIFT-FV	60.3	65.2	58.7	77.2	64.3	75.0	60.1	59.4	61.2	69.8	60.9	68.2	63.5	58.1	7.3	9.5
SIFT-VLAD	60.0	63.9	59.7	77.1	64.9	75.1	60.3	59.8	60.9	69.1	61.6	68.1	63.1	57.5	5.9	8.6

Table 3. The average mAP difference (%) of MLP (Green) and HAE (Blue) on three datasets.

more similar ranking lists, which indicates the rationality of our affinity measurement from the other perspective.

5. Conclusion

In this work, we present the first attempt to investigate visual feature translation, as well as the first attempt at quantifying the affinity among different types of features in visual search. In particular, we propose a Hybrid Auto-Encoder (HAE) to translate visual features. Based on HAE, we design an Undirected Affinity Measurement (UAM) to

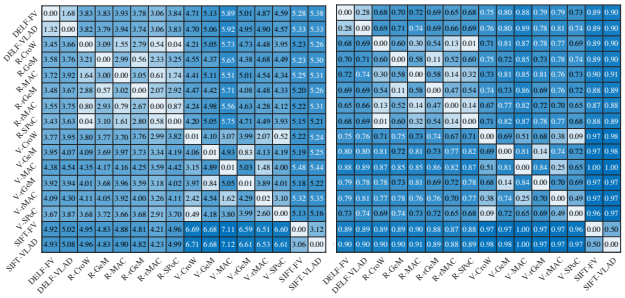

Figure 5. The heat maps of the directed affinity matrix M (left) and the undirected affinity matrix U (right), the values are the averaged results on Holidays, Oxford5k and Paris6k datasets.

quantify the affinity. Extensive experiments have been conducted on several public datasets with 16 different types of widely-used features in visual search. Quantitative results prove the encouraging possibility of feature translation.

Acknowledgments

This work is supported by the National Key R&D Program (No.2017YFC0113000, and No.2016YFB1001503), the Natural Science Foundation of China (No.U1705262, No.61772443, No.61402388 and No.61572410), the Post Doctoral Innovative Talent Support Program under Grant BX201600094, the China Post-Doctoral Science Foundation under Grant 2017M612134, Scientific Research Project of National Language Committee of China (Grant No. YB135-49), and Natural Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

[2] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[3] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *NIPS*, 2007.

[4] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[5] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[8] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

[9] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Co-clustering based classification for out-of-domain documents. In *SIGKDD*, 2007.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[11] Lixin Duan, Ivor W Tsang, and Dong Xu. Domain transfer multiple kernel learning. *PAMI*, 2012.

[12] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *ECCV*, 2016.

[13] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *NIPS*, 2012.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] Jing Huang, S Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *CVPR*, 1997.

[16] Noh Hyeonwoo, Araujo Andre, Sim Jack, Weyand Tobias, and Han Bohyung. Large-scale image retrieval with attentive deep local features. In *ICCV*, 2017.

[17] Hervé Jégou and Ondřej Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *ECCV*, 2012.

[18] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[19] Herve Jegou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[20] Herve Jegou, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. Accurate image search using the contextual dissimilarity measure. *PAMI*, 2010.

[21] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *ECCV*, 2016.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

[23] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *AM MATH SOC*, 1956.

[24] Jingen Liu, Mubarak Shah, Benjamin Kuipers, and Silvio Savarese. Cross-view action recognition via view knowledge transfer. In *CVPR*, 2011.

[25] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[26] Mingsheng Long, Jianmin Wang, Yue Cao, Jiaguang Sun, and S Yu Philip. Deep learning of transferable representation for scalable domain adaptation. *TKDE*, 2016.

[27] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer joint matching for unsupervised domain adaptation. In *CVPR*, 2014.

[28] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

[29] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[30] Jiří Matas Michal Perďoch, Ondřej Chum. Efficient representation of local geometry for large scale object retrieval. In *CVPR*, 2009.

[31] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[32] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. In *CVPRW*, 2015.

[33] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[34] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *TNN*, 2011.

[35] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *TKDE*, 2010.

[36] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[37] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[38] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[39] Danfeng Qin, Stephan Gammeter, Lukas Bossard, Till Quack, and Luc Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR*, 2011.

[40] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018.

[41] Filip Radenović, Giorgos Tolias, and Ondrej Chum. Fine-tuning cnn image retrieval with no human annotation. *PAMI*, 2018.

[42] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, 2007.

[43] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPRW*, 2014.

[44] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *MTA*, 2016.

[45] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. 2011.

[46] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *NIPS*, 2016.

[47] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*.

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[50] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *PAMI*, 2000.

[51] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *ICANN*, 2018.

[52] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR*, 2016.

[53] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[54] Jing Zhang, Wanqing Li, and Philip Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *CVPR*, 2017.

[55] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *PAMI*, 2017.