Journal of Imaging Science and Technology® 60(2): 020402-1–020402-10, 2016.

Hierarchical Manifold Sensing with Foveation and Adaptive Partitioning of the Dataset

Irina Burciu, Thomas Martinetz, and Erhardt Barth
University of Lübeck, Institute for Neuro- and Bioinformatics, Ratzeburger Allee 160, D-23562 Lübeck, Germany
E-mail: irina.burciu@inb.uni-luebeck.de

Abstract. *The authors present a novel method, Hierarchical Manifold Sensing, for adaptive and efficient visual sensing. As opposed to the previously introduced Manifold Sensing algorithm, the new version introduces a way of learning a hierarchical partitioning of the dataset based on k-means clustering. The algorithm can perform on whole images but also on a foveated dataset, where only salient regions are sensed. The authors evaluate the proposed algorithms on the COIL, ALOI, and MNIST datasets. Although they use a very simple nearest-neighbor classifier, on the easier benchmarks, COIL and ALOI, perfect recognition is possible with only six or ten sensing values. Moreover, they show that their sensing scheme yields a better recognition performance than compressive sensing with random projections. On MNIST, state-of-the-art performance cannot be reached, but they show that a large number of test images can be recognized with only very few sensing values. However, for many applications, performance on challenging benchmarks may be less relevant than the simplicity of the solution (processing power, bandwidth) when solving a less challenging problem.* © *2016 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2016.60.2.020402]

INTRODUCTION

We present a novel method called Hierarchical Manifold Sensing (HMS). The objective is to develop appropriate sensing algorithms such as to increase the efficiency of visual sensing by adopting adaptive sensing strategies. The algorithms can also be used for resampling and compression before transmitting a densely sampled signal over a low-bandwidth channel. In other words, we address the question of how to efficiently sample the visual world under the constraint of a limited bandwidth. For example, the bandwidth is limited in human vision by the capacity of the optic nerve and in technical systems by the performance and cost of hardware. As opposed to classical sensing and compression schemes, HMS is based on unsupervised learning, which involves a hierarchical partitioning of the dataset.

Hierarchical Manifold Sensing is inspired by Compressive Sensing (CS).[1] Compressive Sensing is based on the fact that natural images can be encoded sparsely,[2] and thus the number of samples used for representing an image accurately can be reduced by sensing with a random matrix.[3] As opposed to classical sampling, with CS each acquired sensing value is a weighted sum of the original unknown signal. Hierarchical Manifold Sensing works in a similar way, i.e., CS and HMS both make use of a sensing matrix. As opposed to CS, where the sensing matrix does not depend on the sensed data, HMS introduces a two-fold adaptivity: (i) the sensing algorithm adapts to a particular dataset, and (ii) every new sensing value depends on the already acquired sensing values. Thus, sensing in HMS is performed adaptively with optimized weights, and not randomly as in CS.

Schütze et al.[4] presented an alternative adaptive hierarchical sensing (AHS) scheme for efficiently obtaining the sparse coefficients of an image. The sensing process is performed by partially traversing a binary tree and making a measurement at each visited node. The method is adaptive in the sense that after each sensing action, depending on how much gain the sensing operation brings, it is decided whether the entire subtree of the current node is further traversed or whether it is omitted. Adaptive hierarchical sensing was applied on patches of natural images and it was shown that the performance of the method can be improved by choosing an appropriate sparse coding basis and by properly arranging the AHS tree. The results of the method strongly depend on the decision step where a threshold is compared with the measurement values corresponding to the binary tree.

Baraniuk presented a theoretical analysis of CS for manifolds in 2009[5] and showed that, similarly to the theory of CS, only a small number of random linear projections is sufficient to preserve the key information on a signal modeled by a manifold. Later, Chen et al.[6] proposed a statistical framework for CS on manifolds. Their article presents a nonparametric hierarchical Bayesian algorithm that learns a mixture of factor analyzers for manifolds based on the training data. Afterwards, the signal is reconstructed using a limited number of random projections. The method is validated on synthetic and on real datasets, but it is evaluated only for a subset of the MNIST database.

The Manifold Sensing concept was introduced before by Burciu et al. as visual Manifold Sensing (MS),[7] and it was extended afterwards to the foveated version of Manifold Sensing (FMS),[8] which was inspired by the sampling strategy of biological systems. Like Manifold Sensing, HMS is based on a geometric approach. Both MS and FMS are based on learning manifolds of increasing but low dimensionality by using a nonlinear algorithm, namely Locally Linear Embedding (LLE).[9] While sensing, the dataset is continuously adapted and the corresponding embedding is learned. As a further and optional feature, HMS can involve foveation as in the FMS approach.

Received June 30, 2015; accepted for publication Nov. 8, 2015; published online Dec. 10, 2015. Associate Editor: Chunghui Kuo.
1062-3701/2016/60(2)/020402/10/$25.00

Both MS and FMS strongly depend on the choice of the following parameters: (i) the number of neighbors used for LLE, (ii) the decreasing sizes of the adaptive dataset, and (iii) the dimensions of the manifolds at each iteration of the algorithm. Moreover, MS and FMS are operating on the entire dataset while sensing. As highlighted in the FMS article,[8] in a real-time sensing scenario one would need to learn all of the manifolds corresponding to all possible subsets of data before performing the actual sensing. In this article we therefore provide an extended version of MS and FMS, which includes an adequate partitioning learned prior to sensing.

Hierarchical Manifold Sensing as used in this article is also based on learning manifolds of different and low dimensionality. However, for simplicity we here use a linear method Principal Component Analysis (PCA) to learn the low-dimensional representations of the foveated dataset. The hierarchical partitioning of the dataset is performed by clustering the data in the low-dimensional manifolds using the k-means algorithm.

Several approaches aim at developing efficient clustering methods for high-dimensional data; see, for example, Ref. 10. In this work we focus on solving the sensing problem and not on optimizing the approach for hierarchical partitioning of the data. Therefore, we just combine two simple approaches: PCA for dimensionality reduction and k-means for clustering (k-means++ implementation[11]).

Like MS and FMS, HMS is optimized and evaluated with respect to particular recognition tasks and not with respect to the reconstruction error.

In the following section we present the Hierarchical Manifold Sensing method: we first explain how the foveated dataset is created, we then present in detail the steps of hierarchical partitioning of the dataset, and we finally show how the unknown scenes are sensed in a hierarchical way. After that we present the results of this work and conclusions.

HIERARCHICAL MANIFOLD SENSING

Hierarchical Manifold Sensing (HMS) is based on a geometric approach to the problem of efficient sensing. A particular type of environment is represented by the images I^i in a dataset $\mathcal{D} = \{I^1, \ldots, I^p\}$, with p data points of dimension D. In the foveated version of HMS, which is considered here, the dataset $\mathcal{D}$ is first transformed into a foveated dataset $\mathcal{D}_{\text{foveated}}$ that contains only regions of interest out of the original dataset.

The goal is to learn efficient features for classification. This problem is, however, not approached by just unsupervised learning on the whole dataset $\mathcal{D}_{\text{foveated}}$. Instead, a tree structure that involves a hierarchical partitioning of the dataset is learned. The resulting partitioning is used to solve the sensing problem more efficiently, i.e., to use as few sensing actions as possible in order to sense and classify an unknown scene or object.

In the following subsections we first review the procedure of creating the foveated dataset, which was presented in more detail in Ref. 8. Next, we describe the approach for the hierarchical partitioning of the dataset. These two steps define the offline part of the HMS algorithm. After we have learned the foveated hierarchical representation of the given dataset, we can project on it an unknown scene, i.e., a test point outside $\mathcal{D}_{\text{foveated}}$ that we wish to sense. Hierarchical Manifold Sensing thus includes the following main steps.

Creating a Foveated Dataset based on a dataset containing images of known scenes.
Hierarchical Partitioning of the Dataset.
Hierarchical Sensing of Unknown Scenes (here implemented by resampling of unknown test images).

Creating a Foveated Dataset

The foveated dataset $\mathcal{D}_{\text{foveated}}$ contains only the pixels that are salient on average over the dataset. Although these pixels do not necessarily form a compact region of interest (ROI), we will denote the collection of salient pixels as the ROI. The ROI is extracted by using a saliency model based on the geometric invariants of the structure tensor of the images in the dataset $\mathcal{D}$. The invariants of the structure tensor are known to be good predictors of human eye movements for static scenes.[12] In Ref. 12 the properties of the image regions selected by the saccadic eye movements during experiments were analyzed in terms of higher-order statistics. It was shown that image regions with a statistically less redundant structure, such as the ones given by the signals with intrinsic dimension two, contain all the necessary information of a static scene. Therefore, signals with intrinsic dimension two are considered to be more salient. The intrinsic dimension (iD) refers to the relation between the degrees of freedom of a signal domain and the actual degrees of freedom used by a signal. Thus, signals with i0D are constant within a local window, signals with i1D can be approximated by a function of only one variable (e.g., straight lines, edges), and signals with i2D are, for example, corners, curved lines, junctions, and curved edges. In this approach we use the geometric invariant $\boldsymbol{S}$ (determinant) for which the regions of an image with $\boldsymbol{S}$ nonzero are i2D.

Algorithm 1 sketches the steps of creating the foveated dataset. The notations used for the algorithm are included in

Table I. Notations for Algorithm 1.

Notation	Description
$\mathcal{D}$	$\mathcal{D} = \{I^1, \ldots, I^p\}$ contains p data points of dimension D
I^i	Image i with coordinates (x, y)
$\boldsymbol{J}$	Structure tensor
$*, w$	Convolution with kernel w
I_x, I_y	First-order partial derivatives of I^i
$\boldsymbol{S}$	Determinant of the structure tensor
$\boldsymbol{R}$	Average saliency template
$\circ$	Element-wise product of matrixes
T^i	Region of interest for image I^i
$\mathcal{D}_{\text{foveated}}$	$\mathcal{D}_{\text{foveated}} = \{T^1, \ldots, T^p\}$

Table I. Algorithm 1 computes for the given dataset $\mathcal{D}$ the corresponding foveated dataset $\mathcal{D}_{\text{foveated}}$ of the same size p and dimension D; the only difference is that nonsalient pixels are set to zero in every image. For each image I^i in the given dataset $\mathcal{D}$ the geometric invariant S^i is computed as shown in line 4 of the algorithm. Here, S^i is defined as the determinant of the structure tensor $\boldsymbol{J}$[13] (a matrix with the locally averaged products of first-order partial derivatives of image I^i—in line 3). Each invariant image S^i is normalized to the range [0, 1]; the normalized S^i are then summed over all images, and the resulting average saliency map is then transformed into a binary saliency map R based on a threshold θ.

Algorithm 1 Create foveated dataset

1: **function** FOV-DS($\mathcal{D}$)
2: **for all** I^i in $\mathcal{D}$ **do**
3: $\boldsymbol{J} \leftarrow w * \left[I_x^2 \;\; I_x I_y; I_x I_y \;\; I_y^2\right]$
4: $S^i \leftarrow \text{DET}(\boldsymbol{J})$
5: $S^i_{\text{norm}} \leftarrow \text{NORMALIZE}(S^i)$
6: **end for**
7: **if** $\sum_i \left(S^i_{\text{norm}}(x, y)\right) \geq \theta$ **then**
8: $R(x, y) \leftarrow 1$
9: **else**
10: $R(x, y) \leftarrow 0$
11: **end if**
12: **for all** I^i in $\mathcal{D}$ **do**
13: $T^i \leftarrow R \circ I^i$
14: $\mathcal{D}_{\text{foveated}}(i, :) \leftarrow T^i(:)$
15: **end for**
16: **return** $\mathcal{D}_{\text{foveated}}$
17: **end function**

It should be noted that during evaluation, the recognition rate is computed for different values of θ, which results in differently sized regions of interest (different numbers of salient pixels). Based on the training set, an optimal threshold is chosen. By using R, we define for each image the region of interest T^i, i.e., an image that contains the original image values where $R = 1$ and is equal to zero elsewhere.

Hierarchical Partitioning of the Dataset

We create a tree with L levels which contains the hierarchical partitioning of the dataset. The partitioning is performed in the following way: (i) we learn a manifold of dimension N_L (corresponding to level L of the tree) by using PCA and (ii) we cluster the N_L-dimensional representation of the data into k clusters using the k-means algorithm. In the experiments presented here, N_L increases by 1 at each level of the tree. The structure of the tree is presented in Figure 1. The root of the tree is defined as $\mathcal{D}_{11}$, and for $L > 1$, the nodes of the tree are denoted as $\mathcal{D}_{Lk_c}^{(L-1)k_f}$, with $k_c = \overline{1, k}$ being the current cluster and k_f the index of the father node.

Considering the representation of the tree shown in Fig. 1, we initialize the nodes by applying the *partitioning* function presented in Algorithm 2. The notations used in Algorithm 2 are presented in Table II. The *partitioning*

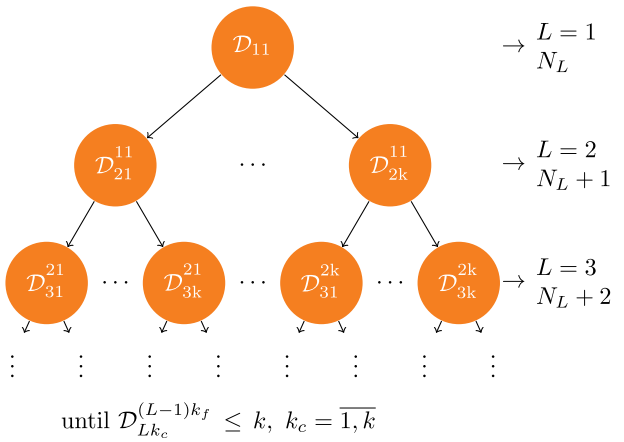


Figure 1. Hierarchical partitioning of the dataset $\mathcal{D}$. The root is defined as $\mathcal{D}_{11}$. Here, L is the level of the tree, k is the number of clusters, k_c is the current cluster, and k_f is the index of the father node. Each node $\mathcal{D}_{lk_c}^{(l-1)k_f}$ of the tree is split into k clusters and contains the learned manifold in dimension N_L, $N_L + 1$, etc. of the corresponding cluster.

function expects as input the current cluster, *currCluster*, the number k of clusters, and N_L, the number of principal components. The function computes for each node of the tree the matrix $\boldsymbol{U}$ which contains the feature vectors of dimensionality N_L (line 3 of the algorithm), the projected data points of the current cluster on the matrix $\boldsymbol{U}$ (line 4 of the algorithm), and the index which indicates to which of the k clusters the images in *currCluster* were clustered with the k-means algorithm (line 5 in the algorithm). We use the *index* to create the subset for the kth child: we select with the *keep* function in line 7 of Algorithm 2 only the images that belong to cluster j. Recursively we compute $\boldsymbol{U}$, Y, *index* and *child* (line 8 of the algorithm). If the current cluster does not have more data points than the number of clusters, the cluster will be empty.

Algorithm 2 Hierarchical partitioning of the dataset

1: **function** PARTITIONING(*currCluster*, k, N_L)
2: **if** size(*currCluster*) $> k$ **then**
3: $\boldsymbol{U} \leftarrow \text{PCA}(\textit{currCluster}, \; N_L)$
4: $Y \leftarrow \boldsymbol{U}' \cdot \textit{currCluster}'$
5: $\textit{index} \leftarrow \text{KMEANS}(Y, \; k)$
6: **for** $j = 1 : k$ **do**
7: $\textit{currCluster.child}(j) \leftarrow \text{KEEP}(\textit{currCluster}, \textit{index}, j)$
8: PARTITIONING($\textit{currCluster.child}(j)$, k, $N_L + 1$)
9: **end for**
10: $\textit{currCluster.hp} \leftarrow \{\boldsymbol{U}, \; Y, \; \textit{index}\}$
11: **else**
12: $\textit{currCluster.hp} \leftarrow \emptyset$
13: **end if**
14: **end function**

Hierarchical Sensing of Unknown Scenes

The HMS algorithm uses the hierarchical partitioning of the dataset presented before to solve in an efficient way the sensing problem. Therefore, an unknown scene, i.e., test

Table II. Notations for Algorithm 2.

Notation	Description
currCluster	Current cluster
L	Current level of the tree
k	Number of clusters for k-means
N_L	Number of principal components for each L
U	Matrix containing the feature vectors, size $(U) = (D \times N_L)$
Y	Projected data points of *currCluster* on matrix U, size $(Y) = (N_L \times p)$
index	$index = \overline{1, k}$, each I_i in *currCluster* belongs to cluster *index*
currCluster.child	Contains the children of the current cluster node
currCluster.hp	Contains U, Y and *index* for each node of the tree

point x_{test} outside $\mathcal{D}$, which we wish to sense, is successively projected on the learned tree. We project x_{test} on the cluster which contains the nearest neighbor of the projected test point on the learned embedding. Algorithm 3 presents this procedure. The notations used are included in Table III. The test point is first projected on the low-dimensional representation of the *rootCluster* in line 5 of the algorithm. In line 6 the function *find1NN* searches for the nearest neighbor *nearNeigh* of the projected point on the learned manifold. The *classify* function checks whether the test point was correctly classified (whether the nearest neighbor and the test point belong to the same class). In line 8 we check to which cluster the nearest neighbor belongs and we define it as *clusterNN*. In line 9 of the algorithm the *currCluster* is updated and it is given by the child that corresponds to cluster *clusterNN*.

Algorithm 3 HMS

```
1:  function PROJECTTESTDATA(rootCluster, x_test)
2:      currCluster ← rootCluster
3:      while currCluster ≠ Ø do
4:          {U, Y, index} ← currCluster.hp
5:          y_test ← U' · x_test'
6:          nearNeigh ← FIND1NN(y_test, Y)
7:          recogHMS ← CLASSIFY(nearNeigh, x_test)
8:          clusterNN ← index(nearNeigh)
9:          currCluster ← currCluster.child(clusterNN)
10:     end while
11: end function
```

The algorithm continues by projecting the test point on the next level L of the tree as long as the current cluster is not empty.

RESULTS

We first evaluated HMS on the Columbia Object Image Library (COIL-20)[14] and Amsterdam Library of Object Images (ALOI)[15] databases. The COIL-20 database contains 1440 grayscale images of 20 objects with 72 images for each object. The images have 128×128 pixels and were taken at object-pose intervals of 5°. The ALOI database is a color image object recognition benchmark with variation in viewing angle. The original dataset contains 1000 different objects with 72 viewing angles for each object. In our experiments we used 50 of all the classes at a quarter resolution 192×144. We worked with resized gray-level images of size 128×128 pixels.

Table III. Notations for Algorithm 3.

Notation	Description
currCluster	Current cluster which contains *currCluster.child* and *currCluster.hp*
x_{test}	Test data point, size $(x_{\text{test}}) = 1 \times D$
y_{test}	Projected x_{test} on the low-dimensional representation
nearNeigh	Nearest neighbor of y_{test} on the low-dimensional representation
recogHMS	Is 1 if x_{test} and *nearNeigh* have the same class and 0 otherwise
clusterNN	Indicates to which cluster the *nearNeigh* belongs

In order to evaluate the presented HMS method with foveation and adapted data partitioning we divided the datasets into training and test data and we computed the recognition rates for the test data by assigning to each test image the corresponding class of the nearest neighbor. For each dataset we chose randomly one image per class and we tested them against the other images that belonged to the training dataset. The goal of HMS is to use as few sensing actions as possible and still obtain the highest possible recognition rate. Therefore, we searched for the minimum number of sensing actions that HMS needs to perform in order to achieve the highest possible recognition rate for all of the tested images.

Performance of HMS With and Without Foveation Versus Random Projections

We explored the benefits of the presented approach by comparing HMS, with and without foveation, with the classical CS method, which uses random projections, i.e., a random Gaussian matrix with rows that have unit length and a smaller number of components. In order to do this, we considered the simplest configuration for the hierarchical partitioning of the data: $k = 2$ clusters and the dimension N_L for the first level of the tree equal to 1. We computed the recognition rate for differently sized regions of interest, e.g., 16% for a foveated dataset and up to 100% for the original dataset. For both HMS and CS we used the same 1NN classifier.

Figure 2 shows, for the database ALOI with 20 classes, how the recognition rate depends on the region of interest, i.e., the number of salient pixels. The curves are plotted for three different numbers (one, three, and six) of sensing values, i.e., for $L = 1$, $L = 2$, and $L = 3$, respectively. For $L = 1$, HMS senses with a sensing matrix of dimension $(N_1 \times D)$, where $N_1 = 1$, i.e., it takes only one sensing value. For $L = 2$, HMS senses with a sensing matrix of $(2 \times D)$, which adds up to $1 + 2 = 3$ sensing values, and so on. It should be noted that for $L = 1$, where only one sensing value is available, foveation deteriorates the result. However,

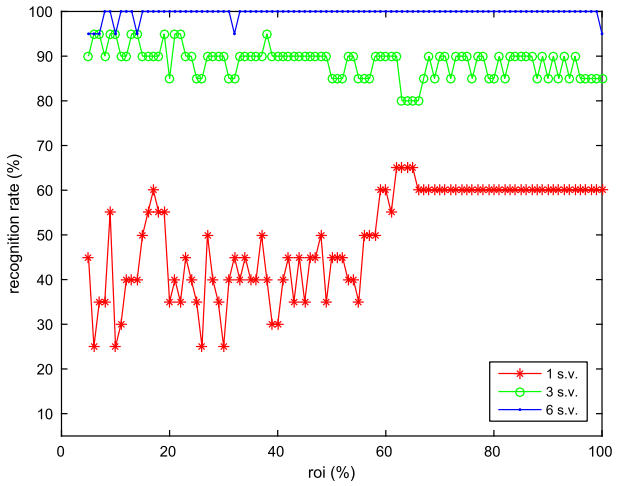


Figure 2. The HMS results for the ALOI 20 database. Recognition rates are shown for different regions of interest with one, three, and six sensing values, respectively. The hierarchical partitioning is performed with $k = 2$ and $N_l = 1$ for $L = 1$.

already with three and six sensing values, the recognition performance does not increase with the number of pixels that are considered, i.e., the performance is equally high with an ROI of only 5% of the image.

Figure 3 shows a selection of seven out of 28 HMS sensing basis functions with an ROI of 8% (first row), 16% (second row), and without foveation (third row). It should be noted that the basis functions for the two different ROIs are specific to the test image shown on the right as each new basis function depends on the previously acquired sensing values. Thus, the basis functions evolve, as we continue sensing, from rather generic to more specific templates. It should also be noted that the ROIs adapt accordingly. In comparison, the third row of Fig. 3 shows the corresponding selection of basis function for the case where the hierarchical partitioning was computed for the original dataset and not for the foveated dataset as shown before. It should be noted that without foveation, the basis functions do also adapt during the hierarchical partitioning but the adaption is less specific.

In Figure 4 (a)–(c) we present representative results with foveation (ROI = 16%) and without foveation (ROI = 100%) for different benchmarks: COIL with 20 classes and ALOI with 20 and 40 classes. We compare the results of HMS with the results obtained by using a random projections matrix for sensing with the corresponding number of sensing values. For all methods we computed 100 runs and we present in Fig. 4 the average of the recognition rate over these runs. As can be seen in Fig. 4 (a)–(c), on the ALOI database with 20 and 40 classes and the COIL-20 database we are able to reach a recognition rate of 100% with a region of interest of only 16% and six sensing values, which corresponds to a compression ratio greater than 4500 in the case of ALOI and greater than 2500 for COIL. The compression ratio is defined by the ratio between the original size of the images of the respective dataset (number of pixels) and the number of sensing values used for recognition. It should be noted that the recognition performance of HMS is higher than the recognition rate obtained with the CS random projections approach on the considered databases.

We conclude that for small databases, such as COIL and ALOI with 20 and 40 classes, it is sufficient to use the HMS algorithm with a simple configuration: a number of two clusters and $N_L = 1$ for $L = 1$. When the database contains more training images, e.g., ALOI with 50 classes, it is worth studying the evolution of HMS for different hierarchical trees, focussing on the different number of principal components of the first level of the tree, N_1, and for a different number of clusters k. We show in Figure 5 the results obtained with HMS for $N_1 = 1, 2$, and 3 in the case of $k = 2$ (a) and $k = 3$ (b). As expected, HMS performs better with a higher number of principal components for the first level of the hierarchical partitioning and with a proper k, considering the number of data points in the training dataset.

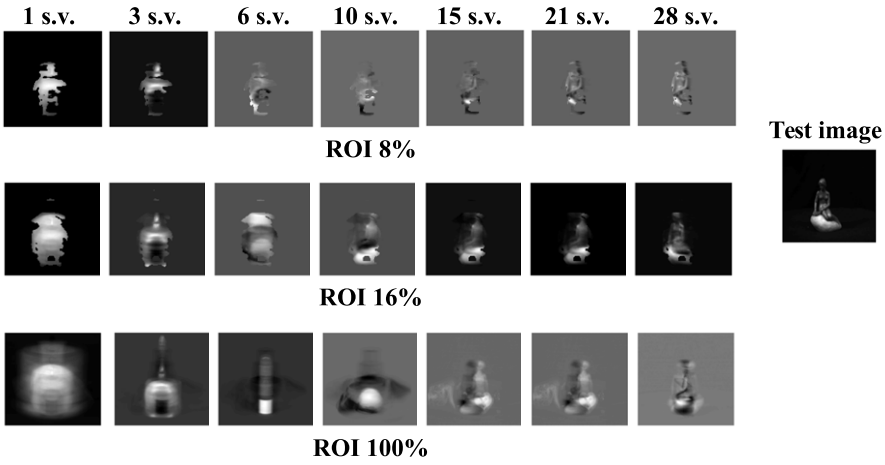
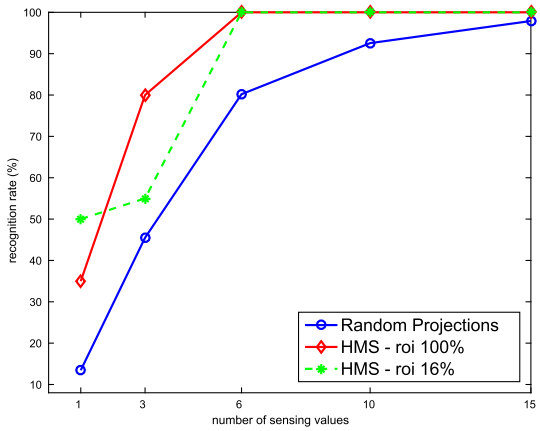
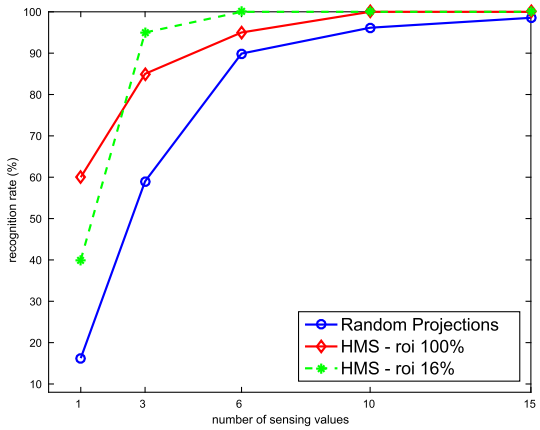


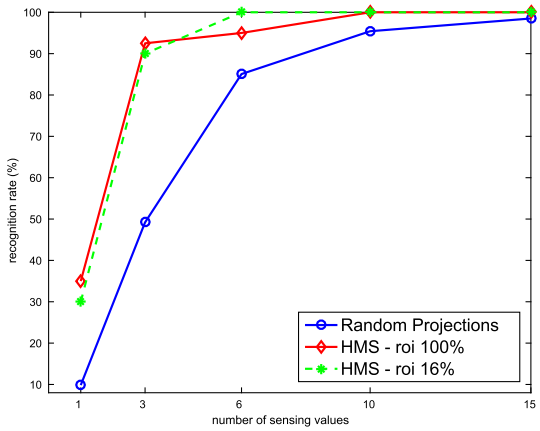
Figure 3. Selected HMS sensing basis functions (seven out of 28 sensing values) with an ROI of 8% (first row), 16% (second row) for the foveated dataset, without foveation (ROI of 100%—third row), and the corresponding test image from ALOI with 20 classes. For the hierarchical partitioning we used $k = 2$ clusters and $N_l = 1$ for $L = 1$.



(a) COIL 20 classes



(b) ALOI 20 classes

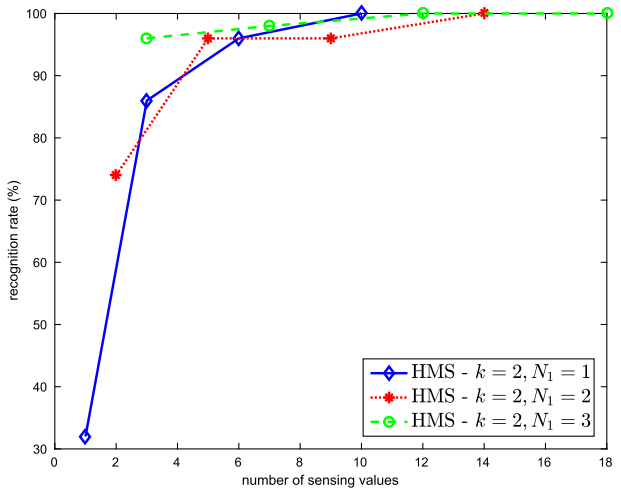


(c) ALOI 40 classes

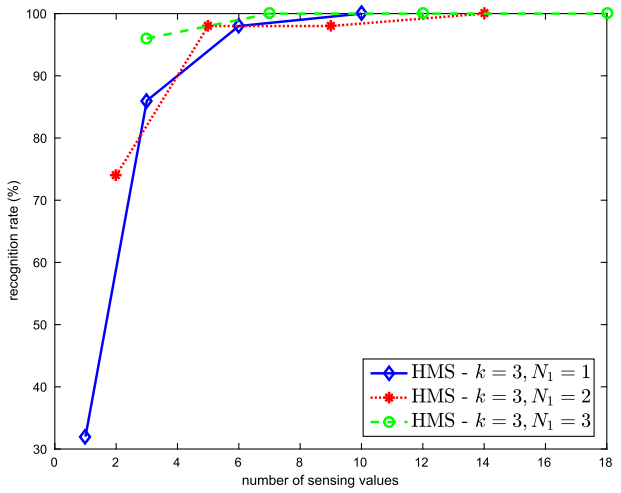
Figure 4. Representative results of HMS with and without foveation versus random projections for different benchmarks.

We also evaluated the algorithm on the highly competitive MNIST[18] benchmark, which consists of handwritten digits from 0 to 9. There are 60,000 images for training and 10,000 for testing.

We first considered the simple configuration for partitioning the data with $N_L = 1$ for $L = 1$ and only $k = 2$ clusters. Although the overall performance of a sensing and recognition scheme with, for example, $L = 12$ is limited to a recognition rate of 93.14%, it is interesting to note that of the 10,000 test images 2491 are already correctly recognized



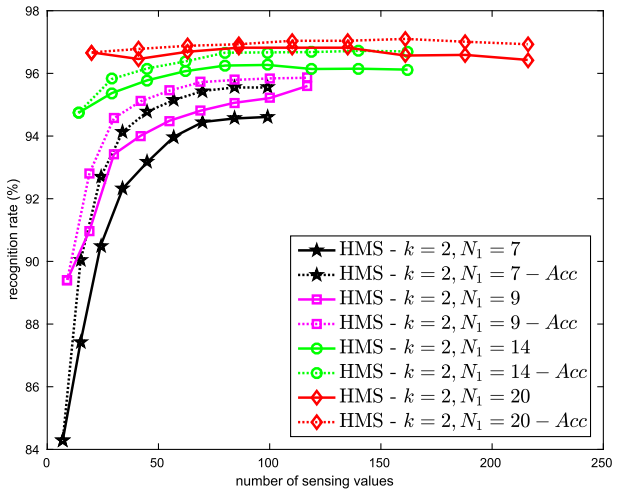
(a) $k = 2$



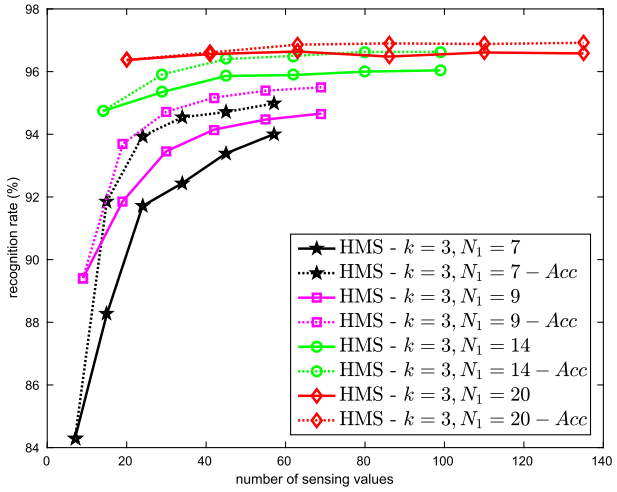
(b) $k = 3$

Figure 5. Results of HMS for different hierarchical partitionings of the training data from ALOI with 50 classes.

with only one sensing action ($L = 1$). Of the remaining test images, 2436 are correctly classified with $L = 2$, 2689 of the remaining images with $L = 3$, and 1290 of the remaining images with $L = 4$. If this scheme is continued up to $L = 12$ and $L = 13$, a total of 98.50% and 98.51%, respectively, of the test images are correctly classified. The difference between 98.50% and 93.14% at $L = 12$ is due to the fact that a few images are obviously misclassified with more sensing values although they would have been correctly classified with fewer. We explored the performance of HMS on the MNIST dataset for different hierarchical partitionings of the training dataset, i.e., with different values of k and N_1. As shown before, for the previously considered databases, the recognition rate grows with N_1. We show in Figure 6 the performance of HMS for different values of N_1 in the case of (a) $k = 2$ and (b) $k = 3$ clusters. The curves are plotted for different numbers of sensing values, i.e., for $L = 1, L = 2, \ldots,$ and $L = 9$ in Fig. 6 (a), and in Fig. 6 (b) for $L = 1, L = 2, \ldots,$ and $L = 6$. As can be seen in Fig. 6 (a), for $k = 2$ and $N_1 = 20$ we reach a recognition rate of 96.69% for $L = 3$, i.e., with 63



(a) $k = 2$



(b) $k = 3$

Figure 6. Results of HMS on MNIST for a hierarchical partitioning of the training data with $N_1 = 7, 9, 14, 20$ and (a) $k = 2$, (b) $k = 3$, and using accumulated sensing values (Acc) over the different levels of the sensing tree.

sensing values and for $L = 4$, we reach a higher recognition rate of 96.82%.

If we accumulate the sensing values over the different levels of the sensing tree the recognition rate improves, as shown in Fig. 6. With $k = 2$ and $N_1 = 20$, HMS reaches a recognition rate of 96.88% with 63 sensing values and 96.93% with 86 sensing values.

In Figure 7 we show the results for HMS compared with CS. The parameters for HMS are $N_1 = 20$ and $k = 2$, and we computed only 10 runs. It should be noted that HMS outperforms CS. The region of interest of 23% seems not to contain sufficient salient pixels in order to reach a higher performance than HMS without foveation, as in the case of the other tested databases.

Figure 8 shows a selection of five sensing basis functions for MNIST obtained with a hierarchical partitioning with $k = 2$ and $N_1 = 9$. In the first column, sample test images with the corresponding number and class are shown, and on each

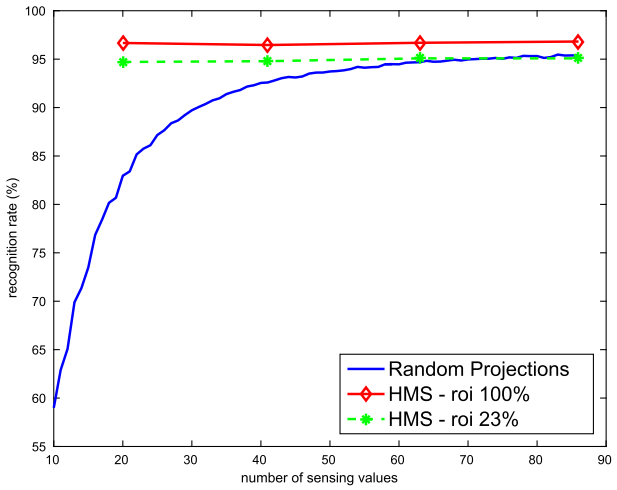


Figure 7. Results on MNIST for HMS versus random projections.

line the corresponding sensing basis functions for a different number of sensing values, i.e., for $L = 1$, $L = 3$, $L = 7$, $L = 8$, and $L = 9$. One can note the evolution from rather generic to more specific templates.

Performance of HMS Versus MS and FMS

In Ref. 7 it has been shown that for the ALOI database with 20 classes, MS needs 38 sensing values in order to reach 100% recognition rate. In Ref. 8, FMS reaches only 65% recognition rate with 15 sensing values. Here, we showed that for a 100% recognition rate only six sensing values are needed for HMS with foveation and ten sensing values for HMS without foveation. Both MS and FMS strongly depend on the number of neighbors selected for the Locally Linear Embedding used to learn the manifolds, on the decreasing size of the adaptive dataset, and on the dimension of the manifolds at each iteration of the algorithm.

The important difference between HMS and MS/FMS is that with HMS the partitioning of the dataset is performed prior to sensing. As a consequence, finding the optimal parameters for the partitioning is more difficult for MS and FMS.

Performance of HMS Versus Other Methods

In 2014 Dornaika et al.[16] developed a semi-supervised feature extraction with an out-of-sample extension algorithm which they applied on a subset of the COIL-20 (18 images from 72 available for each object) database. They randomly selected 50% of the data as the training dataset and the rest as the test dataset. From the training dataset they randomly labeled one, two, and three samples per class and the rest of the data were used as unlabeled data. The data are first preprocessed: PCA is computed in order to preserve 98% of the energy of the dataset. The work Ref. 16 provides a comparison between methods that are based on label propagation and on graph-based semi-supervised embedding. They report the best average classification results on ten random splits for their method for three label samples and for unlabeled (80.4%) and test data (77.4%). They also

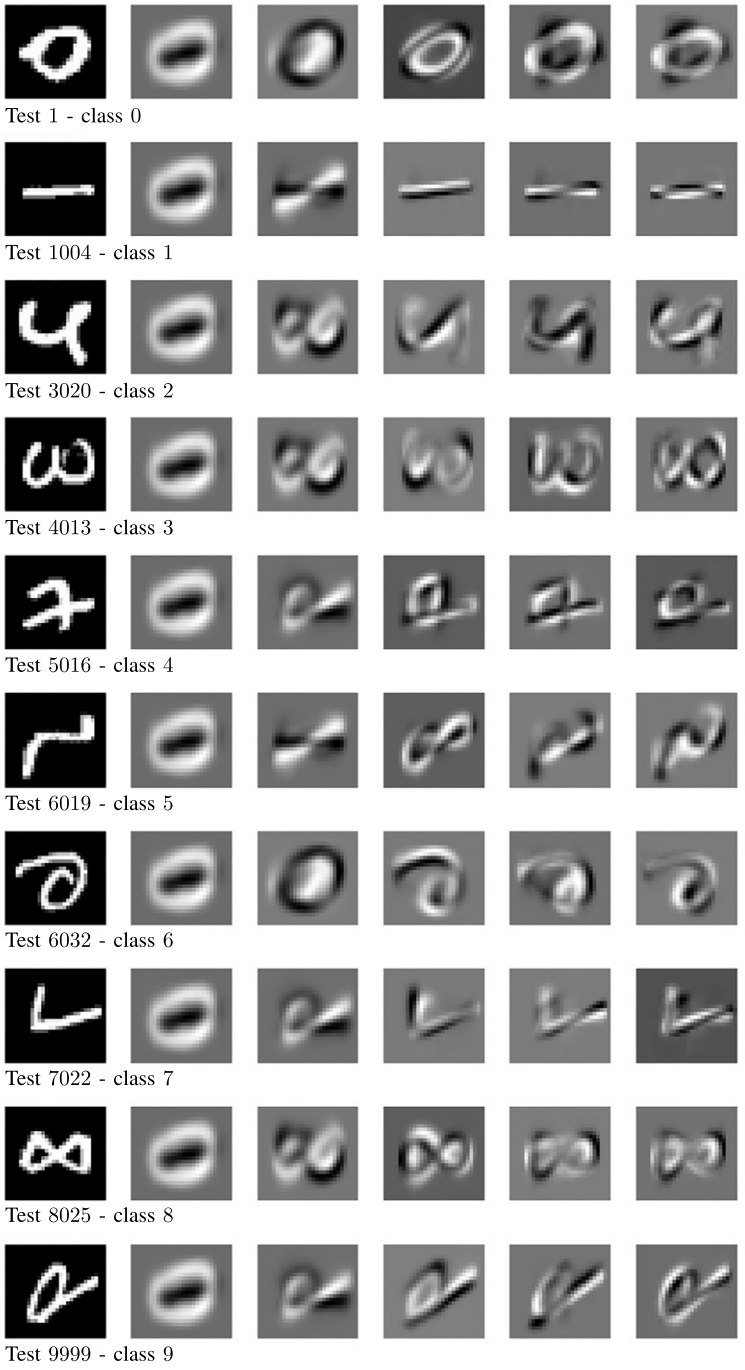


Figure 8. Selected HMS sensing basis functions without foveation (second to fifth column—for $L = 1, 3, 7, 8, 9$) and the corresponding sample test image (first column) from the MNIST dataset. For the hierarchical partitioning we used $k = 2$ clusters and $N_L = 9$ for $L = 1$.

show that when one labeled sample per class is used, their method reaches 61% recognition rate with around 19 feature dimensions. In order to compare HMS with the approach proposed in Ref. 16 we divided the COIL dataset with 72 objects per class into training and test datasets in a similar way to that described before. Although the training dataset consists only of 720 images, HMS performs better than the semi-supervised feature extraction algorithm in Ref. 16. Thus, for a hierarchical partitioning of the training data with $k = 2$ and $N_1 = 1$, HMS reaches an average recognition rate (over ten random splits of the data) of 94.98% with 15 sensing values. If the partitioning is performed with the same number of clusters but $N_1 = 2$, a higher recognition rate of 95.98% at $L = 5$ is reached with 20 sensing values.

A recent article[17] presents an out-of-sample generalization algorithm for supervised manifold learning for classification which is evaluated on the COIL-20 dataset. The authors use 71 images for each of the 20 objects in

COIL, which they normalize, convert to grayscale, and downsample to a resolution of 32×32 pixels. The algorithm embeds the images in a 25-dimensional space. They obtain a minimum average misclassification rate over five runs of approximately 2%. We compared HMS with the approach in Ref. 17 and we obtained an average misclassification rate with ten sensing values of 4.2%, and 100% recognition rate with 15 sensing values. For the hierarchical partitioning we used $k = 2$ and $N_1 = 1$. Thus, HMS reaches 100% recognition rate with even fewer sensing values than in Ref. 17.

On the MNIST database, a baseline method[18] which uses as input to a second-degree polynomial classifier the 40-dimensional feature vector of the training data obtained with PCA has an error rate of 3.3% compared with our 3.32% with 41 sensing values and 3.12% with 63 sensing values, as shown in Fig. 6 (a).

State-of-the-art performance on MNIST has been achieved by a recurrent convolutional neural network (RCNN) approach which introduces recurrent connections into each convolutional layer.[19] The approach reaches a testing error of 0.31% and uses 670,000 parameters. As argued in the following section, our goal is to explore the HMS algorithm in terms of the best recognition rate reached with as few sensing actions as possible, rather than increasing complexity for maximum performance.

CONCLUSIONS AND FUTURE WORK

We have presented a novel algorithm which aims at efficient sensing and have evaluated the efficiency in terms of the resulting recognition performance. We assume the availability of a dataset of images that represent the type of objects and scenes that need to be sensed and recognized. Based on these data, the goal is to learn a sensing strategy such that recognition is possible with few sensing values. Although we use a very simple nearest-neighbor classifier, on easy benchmarks such as COIL and ALOI with only some classes, perfect recognition is possible with only about ten sensing values. On harder benchmarks such as MNIST, state-of-the-art performance could not be reached, but we could show that a large number of test images could be recognized with only very few sensing values. Such performance resembles human performance, since humans can effortlessly recognize a multitude of objects based on just the gist of a scene, and require scrutiny for less familiar objects and more difficult recognition tasks. A further bio-inspired element of our algorithm is foveation, and we have shown that gist-like sensing and recognition requires the whole image, whereas more refined sensing can be reduced to only few salient locations without deteriorating recognition performance. It should be noted, however, that we do not use the saliency of the actual image but only the average saliency of all images in the hierarchical dataset. In terms of compressive sensing, we have here proposed a sensing scheme with a learned sensing matrix, and we have shown that it leads to better recognition performance than random sensing. In the case of foveation, the sensing matrix is also sparse in addition to being learned, i.e., adapted to the specific dataset.

A weakness of the proposed approach is that it offers a rather high number of choices. For example, it is not obvious in which dimension to start sensing and how to increase the dimension of the embedding manifolds. In future work, however, this weakness could be turned into a benefit by exploring different strategies. A further weakness is that the method, like many others, will be less efficient if many different objects need to be recognized in the same scene. This scenario would most likely require a preliminary stage of object detection and rough object segmentation.

It should be noted that we are here addressing a problem that is not typically addressed in current computer-vision challenges but is becoming increasingly relevant as computer-vision systems are becoming more pervasive. While currently the focus is on maximizing recognition performance, an equally challenging problem consists of finding the simplest solution for a given problem. Simplicity can be defined in different ways, and we here adopt the approach of using a minimum number of sensing values and a simple classifier. This reduces both the required bandwidth of the sensor and the required processing power.

ACKNOWLEDGMENTS

This research is funded by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/1].

REFERENCES

[1] E. Candès and M. Wakin, "Introduction to compressive sampling," IEEE Signal Process. Mag. **25**, 21–30 (2008).
[2] B. Olshausen and D. Field, "Natural image statistics and efficient coding," Netw., Comput. Neural Syst. **7**, 333–339 (1996).
[3] D. Donoho, "Compressed sensing," IEEE Trans. Inf. Theory **52**, 1289–1306 (2006).
[4] H. Schütze, E. Barth, and T. Martinetz, "An adaptive hierarchical sensing scheme for sparse signals," Proc. SPIE **9014**, 15:1–8 (2014).
[5] R. Baraniuk and M. Wakin, "Random projections of smooth manifolds," Found. Comput. Math. **9**, 51–77 (2009).
[6] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin, "Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: algorithm and performance bounds," IEEE Trans. Signal Process. **58**, 6140–6155 (2010).
[7] I. Burciu, A. Ion-Mărgineanu, T. Martinetz, and E. Barth, "Visual manifold sensing," Proc. SPIE **9014**, 48:1–8 (2014).
[8] I. Burciu, T. Martinetz, and E. Barth, "Foveated manifold sensing for object recognition," *Proc. IEEE Black Sea Conf. on Communications and Networking* (2015), pp. 196–200.
[9] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science **290**, 2323–2326 (2000).
[10] N. Tajunisha and V. Saravanan, "An efficient method to improve the clustering performance for high dimensional data by Principal Component Analysis and modified K-means," International Journal of Database Management Systems **3**, (2011).
[11] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," *SODA '07: Proc. Eighteenth Annual ACM-SIAM symposium on Discrete Algorithms* (2007), pp. 1027–1035.
[12] C. Zetzsche, K. Schill, H. Deubel, G. Krieger, E. Umkehrer, and S. Beinlich, "Investigation of a sensorimotor system for saccadic scene analysis: an integrated approach," *From Animals to Animats 5: Proc. Fifth Int. Conf. on Simulation of Adaptive Behavior*, edited by R. Pfeifer *et al.* (MIT Press, 1998), Vol. 5, pp. 120–126.
[13] B. Jähne, H. Haußecker, and P. Geißler, *Handbook of Computer Vision and Applications* (Academic Press, 1999), Vol. 2.

[14] S. A. Nene, S. K. Nayar, and H. Murase, Columbia Object Image Library (COIL-20), *Technical Report* CUCS-005-96 (1996).
[15] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The Amsterdam Library of Object Images," Int. J. Comput. Vis. **61**, 103–112 (2005).
[16] F. Dornaika, Y. El. Traboulsi, B. Cases, and A. Assoum, "Image classification via semi-supervised feature extraction with out-of-sample extension," *Advances in Visual Computing ISCV Part I* (2014), Vol. 8887, pp. 182–192.
[17] E. Vural and C. Guillemot, Out-of-sample generalizations for supervised manifold learning for classification, http://arxiv.org/abs/1502.02410 [cs.CV], (2015).
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE **86**, 2278–2324 (1998).
[19] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2015).