Mean Shift Based Clustering in High Dimensions: A Texture Classification Example

Bogdan Georgescu(1) Ilan Shimshoni(3) Peter Meer(1,2)

Computer Science (1)
Electrical and Computer Engineering(2)
Rutgers University, Piscataway, NJ 08854, USA
georgesc, meer@caip.rutgers.edu

Industrial Engineering and Management(3)
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL
ilans@ie.technion.ac.il

Abstract

Feature space analysis is the main module in many computer vision tasks. The most popular technique, k-means clustering, however, has two inherent limitations: the clusters are constrained to be spherically symmetric and their number has to be known a priori. In nonparametric clustering methods, like the one based on mean shift, these limitations are eliminated but the amount of computation becomes prohibitively large as the dimension of the space increases. We exploit a recently proposed approximation technique, locality-sensitive hashing (LSH), to reduce the computational complexity of adaptive mean shift. In our implementation of LSH the optimal parameters of the data structure are determined by a pilot learning procedure, and the partitions are data driven. As an application, the performance of mode and k-means based textons are compared in a texture classification study.

1. Introduction

Representation of visual information through feature space analysis received renewed interest in recent years, motivated by content based image retrieval applications. The increase in the available computational power allows today the handling of feature spaces which are high dimensional and contain millions of data points.

The structure of high dimensional spaces, however, defies our three dimensional geometric intuition. Such spaces are extremely sparse with the data points far away from each other [17, Sec.4.5.1]. Thus, to infer about the local structure of the space only a small number of data points may be available, which can yield erroneous results. The phenomenon is known in the statistical literature as the *curse of dimensionality*, and its effect increases exponentially with the dimension. The curse of dimensionality can be avoided only by imposing a fully parametric model over the data [6, p.203], an approach which is not feasible for a high dimensional feature space with a complex structure.

The goal of feature space analysis is to reduce the data to a few significant features through a procedure known under many different names, clustering, unsupervised learning, or vector quantization. Most often different variants of *k-means clustering* are employed, in which the feature space is represented as a mixture of normal distributions [6, Sec.10.4.3]. The number of mixture components k is usually set by the user.

The popularity of the k-means algorithm is due to its low computational complexity of $O(nkNd)$, where n is the number of data points, d the dimension of the space, and N the number of iterations which is always small relative to n. However, since it imposes a rigid delineation over the feature space and requires a reasonable guess for the number of clusters present, the k-means clustering can return erroneous results when the embedded assumptions are not satisfied. Moreover, the k-means algorithm is not robust, points which do not belong to any of the k clusters can move the estimated means away from the densest regions.

A robust clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters, is the *mean shift* based clustering. This is also an iterative technique, but instead of the means, it estimates the modes of the multivariate distribution underlying the feature space. The number of clusters is obtained automatically by finding the centers of the densest regions in the space (the modes). See [1] for details. Under its original implementation the mean shift based clustering cannot be used in high dimensional spaces. Already for $d = 7$, in a video sequence segmentation application, a fine-to-coarse hierarchical approach had to be introduced [5].

The most expensive operation of the mean shift method is finding the closest neighbors of a point in the space. The problem is known in computational geometry as *multidimensional range searching* [4, Chap.5]. The goal of the range searching algorithms is to represent the data in a structure in which proximity relations can be determined in less than $O(n)$ time. One of the most popular structures, the kD-tree, is built in $O(n \log n)$ operations, where the proportionality constant increases with the dimension of the space. A query selects the points within a rectangular region delimited by an interval on each coordinate axis, and the query time for kD-trees has complexity bounded by $O\left(n^{\frac{d-1}{d}}+m\right)$, where m is the number of points found. Thus, for high dimensions the complexity of a query is practically linear, yielding the *computational curse of dimensionality*. Recently, several probabilistic algorithms have been proposed for approximate nearest neighbor search. The algorithms yield sublinear complexity with a speedup which depends on the desired accuracy [7, 10, 11].

In this paper we have adapted the algorithm in [7] for mean shift based clustering in high dimensions. Working with data in high dimensions also required that we extend the adaptive mean shift procedure introduced in [2]. All computer vision applications of mean shift until now, such as image segmentation, object recognition and tracking, were in relatively low-dimensional spaces. Our implementation opens the door to use mean shift in tasks based on high-dimensional features.

In Section 2 we present a short review of the adaptive mean-shift technique. Locality-sensitive hashing, the technique for approximate nearest neighbor search is described in Section 3, where we have also introduced refinements to handle data with complex structure. In Section 4 the performance of adaptive mean shift (AMS) in high dimensions is investigated, and in Section 5 AMS is used for texture classification based on textons.

2. Adaptive Mean Shift

Here we only review some of the results described in [2] which should be consulted for the details.

Assume that each data point $\mathbf{x}_i \in \mathcal{R}^d$, $i = 1, \ldots, n$ is associated with a bandwidth value $h_i > 0$. The *sample point* estimator

$$\hat{f}_K(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}\,k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) \tag{1}$$

based on a spherically symmetric kernel K with bounded support satisfying

$$K(\mathbf{x}) = c_{k,d}\,k(\|\mathbf{x}\|^2) > 0 \qquad \|\mathbf{x}\| \le 1 \tag{2}$$

is an adaptive nonparametric estimator of the density at location $\mathbf{x}$ in the feature space. The function $k(x)$, $0 \le x \le 1$, is called the *profile* of the kernel, and the normalization constant $c_{k,d}$ assures that $K(\mathbf{x})$ integrates to one. The function $g(x) = -k'(x)$ can always be defined when the derivative of the kernel profile $k(x)$ exists. Using $g(x)$ as the profile, the kernel $G(\mathbf{x})$ is defined as $G(\mathbf{x}) = c_{g,d}\,g(\|\mathbf{x}\|^2)$.

By taking the gradient of (1) the following property can be proven

$$\mathbf{m}_G(\mathbf{x}) = C\frac{\hat{\nabla} f_K(\mathbf{x})}{\hat{f}_G(\mathbf{x})} \tag{3}$$

where C is a positive constant and

$$\mathbf{m}_G(\mathbf{x}) = \frac{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}\mathbf{x}_i\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)} - \mathbf{x} \tag{4}$$

is called the *mean shift vector*. The expression (3) shows that at location $\mathbf{x}$ the weighted mean of the data points selected with kernel G is proportional to the normalized density gradient estimate obtained with kernel K. The mean shift vector thus points toward the direction of maximum increase in the density. The implication of the mean shift property is that the iterative procedure

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n}\frac{\mathbf{x}_i}{h_i^{d+2}}\,g\left(\left\|\frac{\mathbf{y}_j-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}\,g\left(\left\|\frac{\mathbf{y}_j-\mathbf{x}_i}{h_i}\right\|^2\right)} \quad j = 1, 2, \ldots \tag{5}$$

is a hill climbing technique to the nearest stationary point of the density, i.e., a point in which the density gradient vanishes. The initial position of the kernel, the starting point of the procedure $\mathbf{y}_1$ can be chosen as one of the data points $\mathbf{x}_i$. Most often the points of convergence of the iterative procedure are the modes (local maxima) of the density.

There are numerous methods described in the statistical literature to define h_i, the bandwidth values associated with the data points, most of which use a pilot density estimate [17, Sec.5.3.1]. The simplest way to obtain the pilot density estimate is by nearest neighbors [6, Sec.4.5]. Let $\mathbf{x}_{i,k}$ be the k-nearest neighbor of the point $\mathbf{x}_i$. Then, we take

$$h_i = \|\mathbf{x}_i - \mathbf{x}_{i,k}\|_1 \tag{6}$$

where L_1 norm is used since it is the most suitable for the data structure to be introduced in the next section. The choice of the norm does not have a major effect on the performance. The number of neighbors k should be chosen large enough to assure that there is an increase in density within the support of most kernels having bandwidths h_i. While the value of k should increase with d the dimension of the feature space, the dependence is not critical for the performance of the mean shift procedure, as will be seen in Section 4. When all $h_i = h$, i.e., a single global bandwidth value is used, the adaptive mean shift (AMS) procedure becomes the fixed bandwidth mean shift (MS) discussed in [1].

A robust nonparametric clustering of the data is achieved by applying the mean shift procedure to a representative subset of the data points. After convergence, the detected modes are the cluster centers, and the shape of the clusters is determined by the basins of attraction. See [1] for details.

3. Locality-Sensitive Hashing

The bottleneck of mean shift in high dimensions is the need for a fast algorithm to perform neighborhood queries when computing (5). The problem has been addressed before in the vision community by sorting the data according to each of the d coordinates [13], but a significant speedup was achieved only when the data is close to a low-dimensional manifold.

Recently new algorithms using tools from probabilistic approximation theory were suggested for performing approximate nearest neighbor search in high dimensions for general datasets [10, 11] and for clustering data [9, 14]. We use the approximate nearest neighbor algorithm based on *locality-sensitive hashing* (LSH) [7] and adapted it to handle the complex data met in computer vision applications. In a task of estimating the pose of articulated objects, described in these proceedings [16], the LSH technique was extended to accommodate distances in the parameter space.

3.1. High Dimensional Neighborhood Queries

Given n points in $\mathcal{R}^d$ the mean shift iterations (5) require a neighborhood query around the current location $\mathbf{y}_j$. The naive method is to scan the whole dataset and test whether the kernel of the point $\mathbf{x}_i$ covers $\mathbf{y}_j$. Thus, for each mean computation the complexity is $O(nd)$. Assuming that for every point in the dataset this operation is performed N times (a value which depends on the h_i's and the distribution of the data), the complexity of the mean shift algorithm is $O(n^2dN)$.

To improve the efficiency of the neighborhood queries the following data structure is constructed. The data is tessellated L times with random partitions, each defined by K inequalities (Figure 1). In each partition K pairs of random numbers, d_k and v_k are used. First d_k, an integer between 1 and d is chosen, followed by v_k, a value within the range of the data along the d_k-th coordinate.

The pair (d_k, v_k) partitions the data according to the inequality

$$x_{i,d_k} \leq v_k \qquad i = 1, \ldots, n \tag{7}$$

where x_{i,d_k} is the selected coordinate for the data point $\mathbf{x}_i$. Thus, for each point $\mathbf{x}_i$ each partition yields a K-dimensional boolean vector (inequality true/false). Points which have the same vector lie in the same cell of the partition. Using a hash function, all the points belonging to the same cell are placed in the same bucket of a hash table. As we have L such partitions, each point belongs simultaneously to L cells (hash table buckets).

To find the neighborhood of radius h around a query point $\boldsymbol{q}$, L boolean vectors are computed using (7). These vectors index L cells C_l, $l = 1, \ldots, L$ in the hash table. The points in their union $C_\cup = \bigcup_{l=1}^{L} C_l$ are the ones returned by the query (Figure 1). Note that any $\boldsymbol{q}$ in the intersection $C_\cap = \bigcap_{l=1}^{L} C_l$ will return the same result. Thus $C_\cap$ determines the resolution of the data structure, whereas $C_\cup$ determines the set of the points returned by the query. The described technique is called *locality-sensitive hashing* (LSH) and was introduced in [10].

Points close in $\mathcal{R}^d$ have a higher probability for collision in the hash table. Since $C_\cap$ lies close to the center of $C_\cup$, the query will return most of the nearest neighbors of $\boldsymbol{q}$. The example in Figure 1 illustrates the approximate nature of the query. Parts of an L_1 neighborhood centered on $\boldsymbol{q}$ are not covered by $C_\cup$ which has a different shape. The approximation errors can be reduced by building data structures with larger $C_\cup$'s, however, this will increase the running time of a query.

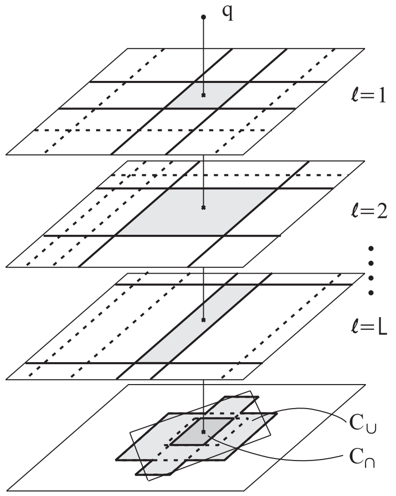


Figure 1: The locality-sensitive hashing data structure. For the query point $\boldsymbol{q}$ the overlap of L cells yields the region $C_\cup$ which approximates the desired neighborhood.

3.2. Optimal Selection of K and L

The values for K and L determine the expected volumes of $C_\cap$ and $C_\cup$. The average number of inequalities used for each coordinate is K/d, partitioning the data into $K/d + 1$ regions. The average number of points N_{C_l} in a cell C_l and $N_{C_\cup}$ in their union $C_\cup$ is

$$N_{C_l} \approx n\,(K/d + 1)^{-d} \quad N_{C_\cup} \approx L\,N_{C_l}\,. \tag{8}$$

Note that in estimating $N_{C_\cup}$ we disregard that the points belong to several C_l's.

Qualitatively, the larger the value for K, the number of cuts in a partition, the smaller the average volume of the cells C_l. Similarly, as the number of partitions L increases, the volume of $C_\cap$ decreases and of $C_\cup$ increases. For a given K, only values of L below a certain bound are of interest. Indeed, once L exceeds this bound all the neighborhood of radius h around $\boldsymbol{q}$ has been already covered by $C_{\cup}$. Thus, larger values of L will only increase the query time with no improvement in the quality of the results.

The optimal values of K and L can be derived from the data. A subset of data points $\mathbf{x}_j, j = 1, \cdots, m \ll n$, is selected by random sampling. For each of these data points, the L_1 distance h_j (6) to its k-nearest neighbor is determined accurately by the traditional linear algorithm.

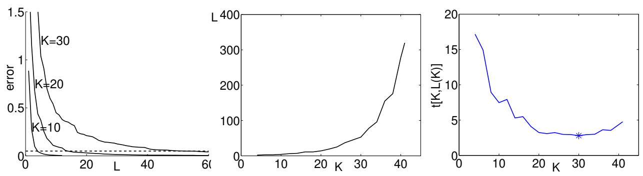
In the approximate nearest neighbor algorithm based on LSH, for any pair of K and L, we define for each of the m points $h_j^{(K,L)}$, the distance to the k-nearest neighbor returned by the query. When the query does not return the correct k-nearest neighbors $h_j^{(K,L)} > h_j$. The total running time of the m queries is $t(K, L)$. The optimal (K, L) is then chosen such that

$$(K, L) = \arg\min_{K,L} t(K, L) \tag{9}$$

$$\text{subject to:} \quad \frac{1}{m}\sum_{j=1}^{m} \frac{h_j^{(K,L)}}{h_j} \leq (1 + \epsilon)$$

where ϵ is the LSH approximation threshold set by the user.

The optimization is performed as a numerical search procedure. For a given K we compute, as a function of L, the approximation error of the m queries. This is shown in Figure 2a for a thirteen-dimensional real data set. By thresholding the family of graphs at $\epsilon = 0.05$, the function $L(K)$ is obtained (Figure 2b). The running time can now be expressed as $t[K, L(K)]$, i.e., a one-dimensional function in K, the number of employed cuts (Figure 2c). Its minimum is K_{min} which together with $L(K_{min})$ are the optimal parameters of the LSH data structure.



(a) (b) (c)

Figure 2: Determining the optimal K and L. (a) Dependence of the approximation error on L for $K = 10, 20, 30$. The curves are thresholded at $\epsilon = 0.05$, dashed line. (b) Dependence of L on K for $\epsilon = 0.05$. (c) The running time $t[K, L(K))]$. The minimum is marked '$*$'.

The family of error curves can be efficiently generated. The number of partitions L is bounded by the available computer memory. Let L_{max} be that bound. Similarly, we can set a maximum on the number of cuts, K_{max}. Next, the LSH data structure is built with (K_{max}, L_{max}). Then, the approximation error is computed incrementally for $L = 1, \cdots, L_{max}$ by adding one partition at a time. This yields $L(K_{max})$ which is subsequently used as L_{max} for $K_{max} - 1$, etc.

3.3. Data Driven Partitions

The strategy of generating the L random tessellations has an important influence on the performance of locality-sensitive hashing. In [7] the coordinates d_k have equal chance to be selected and the values v_k are uniformly distributed over the range of the corresponding coordinate. This partitioning strategy works well only when the density of the data is approximately uniform in the entire space. However, feature spaces associated with vision applications are often multimodal. In [10, 11] the problem of nonuniformly distributed data was dealt with by building several data structures with different values of K and L to accommodate the different local densities. The query is performed first under the assumption of a high density, and when it fails it is repeated for lower densities. The process terminates when the nearest neighbors are found.

Our approach is to sample according to the marginal distributions along each coordinate. We use a few points $\mathbf{x}_i$ chosen at random from the data set. For each point one of its coordinates is selected at random to define a cut. Using more than one coordinate from a point would imply sampling from partial joint densities, but does not seem to be more advantageous. Our adaptive, data driven strategy assures that in denser regions more cuts will be made yielding smaller cells, while in sparser regions there will be less cuts. On average all cells will contain a similar number of points.

The two-dimensional data in Figure 3b and 3b comprised of four clusters and uniformly distributed background, is used to demonstrate the two sampling strategies. In both cases the same number of cuts were used but the data driven method places most of the cuts over the clusters (Figure 3b). For a quantitative performance assessment a data set of ten normal distributions with arbitrary shapes (5000 points each) were defined in 50 dimensions. When the data driven strategy is used, the distribution of the number of points in a cell is much more compact and their average value is much lower (Figure 3c). As a consequence, the data driven strategy yields more efficient k-nearest neighbor queries for complex data sets.

4. Mean Shift in High Dimensions

Given $\mathbf{y}_j$, the current location in the iterations, an LSH based query retrieves the approximate set of neighbors needed to compute the next location (5). The resolution of the data analysis is controlled by the user. In the fixed bandwidth MS method the user provides the bandwidth parameter h. In the AMS method, the user sets the number

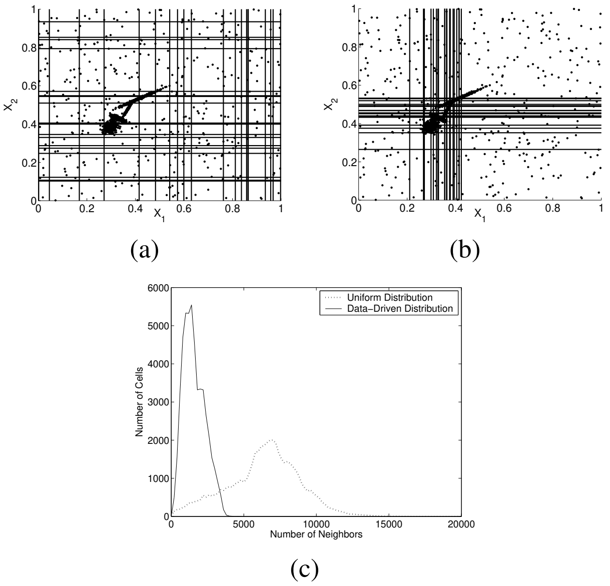


Figure 3: Uniform vs. data driven partitions. Typical result for two-dimensional data obtained with (a) uniform; (b) data driven strategy. (c) Distribution of points-per-cell for a 50-dimensional data set.

of neighbors k used in the pilot density procedure. The parameters K and L of the LSH data structure are selected employing the technique discussed in Section 3.2. The bandwidths h_i associated with the data points are obtained by performing n neighborhood queries. Once the bandwidths are set, the adaptive mean shift procedure runs at approximately the same cost as the fixed bandwidth mean shift. Thus, the difference between MS and AMS is only one additional query per point.

An ad-hoc procedure provides further speedup. Since the resolution of the data structure is $C_\cap$, with high probability one can assume that all the points within $C_\cap$ will converge to the same mode. Thus, once any point from a $C_\cap$ is associated with a mode, the subsequent queries to $C_\cap$ automatically return this mode and the mean shift iterations stop. The modes are stored in a separate hash table whose keys are the L boolean vectors associated with $C_\cap$.

4.1. Adaptive vs. Fixed Bandwidth Mean Shift

To illustrate the advantage of adaptive mean shift, a data set containing 125,000 points in a 50-dimensional cube was generated. From these $10 \times 2{,}500$ points belonged to ten spherical normal distributions (clusters) whose means were positioned on a line through the origin. The standard deviation increases as the mean becomes more distant from the origin. For an adjacent pair of clusters, the ratio of the sum of standard deviations to the distance between the means was kept constant. The remaining 100,000 points were uniformly distributed in the 50-dimensional cube. Plotting the distances of the data points from the origin yields a graph very similar to the one in Figure 4a. Note that the points farther from the origin have a larger spread.

The performance of the fixed bandwidth (MS) and the adaptive mean shift (AMS) procedures is compared for various parameter values in Figure 4. The experiments were performed for 500 points chosen at random from each cluster, a total of 5000 points. The location associated with each selected point *after* the mean shift procedure, is the employed performance measure. Ideally this location should be near the center of the cluster to which the point belongs.

In the MS strategy, when the bandwidth h is small, due to the sparseness of the high-dimensional space very few points have neighbors within distance h. The mean shift procedure does not start and the allocation of the points is to themselves (Figure 4a). On the other hand as h increases the windows become too large for some of the local structures and points may converge incorrectly to the center (mode) of an adjacent cluster (Figures 4b to 4d).

The pilot density estimation in the AMS strategy automatically adapts the bandwidth to the local structure. The parameter k, the number of neighbors used for the pilot estimation does not have a strong influence. The data is processed correctly for $k = 100$ to 500, except for a few points (Figures 4e to 4g), and even for $k = 700$ only some of the points in the cluster with the largest spread converge to the adjacent mode (Figure 4h). The superiority of the adaptive mean shift in high dimensions is clearly visible. Due to the sparseness of the 50-dimensional space, the 100,000 points in the background did not interfere with the mean shift processes under either strategy, proving its robustness.

The use of the LSH data structure in the mean shift procedure assures a significant speedup. We have derived four different features spaces from a texture image with the filter banks discussed in the next section. The spaces had dimension $d = 4, 8, 13$ and 48, and contained $n = 65536$ points. An AMS procedure was run both with linear and approximate queries for 1638 points. The number of neighbors in the pilot density estimation was $k = 100$. The approximation error of the LSH was $\epsilon = 0.05$. The running times (in seconds) in Table 1 show the achieved speedups.

Table 1: Running Times of AMS Implementations

d	Traditional	LSH	Speedup
4	1,507	80	18.8
8	1,888	206	9.2
13	2,546	110	23.1
48	5,877	276	21.3

The speedup will increase with the number of data points n, and will decrease with the number of neighbors k. Therefore in the mean shift procedure the speedup is not as high as in applications in which only a small number of neighbors are required.

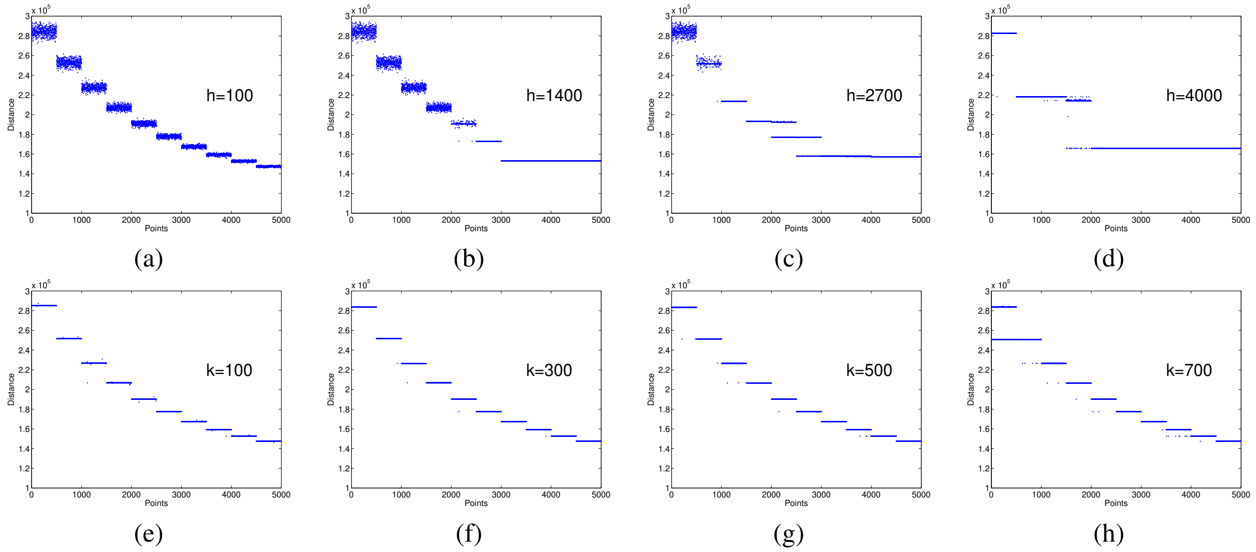


Figure 4: Distance from the origin of 5000 points from ten 50-dimensional clusters *after* fixed bandwidth mean shift (MS): (a) to (d); and adaptive mean shift (AMS) : (e) to (h). The parameters: MS – bandwidth h; AMS – number of neighbors k.

5. Texture Classification

Efficient methods exist for texture classification under varying illumination and viewing direction [3],[12], [15], [18]. In the state-of-the-art approaches a texture is characterized through *textons*, which are cluster centers in a feature space derived from the input. Following [12] this feature space is built from the output of a filter bank applied at every pixel. However, as was shown recently [19], neighborhood information in the spatial domain may also suffice.

The approaches differ in the employed filter bank.

- **LM**: A combination of 48 anisotropic and isotropic filters were used by Leung and Malik [12] and Cula and Dana [3]. The filters are Gaussian masks, their first derivative and Laplacian, defined at three scales. Because of the oriented filters, the representation is sensitive to texture rotations. The feature space is 48 dimensional.
- **S**: A set of 13 circular symmetric filters was used by Schmid [15] to obtain a rotationally invariant feature set. The feature space is 13 dimensional.
- **M4**, **M8**: Both representations were proposed by Varma and Zissermann [18]. The first one (M4) is based on 2 rotationally symmetric and 12 oriented filters. The second set is an extension of the first one at 3 different scales. The feature vector is computed by retaining only the maximum response for the oriented filters (2 out of 12 for M4 and 6 out of 36 for M8), thus reducing the dependence on the global texture orientation. The feature space is 4 respectively 8 dimensional.

To find the textons, usually the standard k-means clustering algorithm is used, which as was discussed in Section 1 has several limitations. The shape of the clusters is restricted to be spherical and their number has to be set prior to the processing.

The most significant textons are aggregated into the *texton library*. This serves as a dictionary of representative local structural features and must be general enough to characterize a large variety of texture classes. A texture is then modeled through its *texton histogram*. The histogram is computed by defining at every pixel a feature vector, replacing it with the closest texton from the library (vector quantization) and accumulating the results over the entire image.

Let two textures i and j be characterized by the histograms H_i and H_j built from T textons. As in [12] the χ^2 distance between these two texton distributions

$$\chi^2(H_i, H_j) = \sum_{t=1}^{T} \frac{[H_i(t) - H_j(t)]^2}{H_i(t) + H_j(t)} \qquad (10)$$

is used to measure similarity, although note the absence of the factor $1/2$ to take into account that the comparison is between *two* histograms derived from data. In a texture classification task the training image with the smallest distance from the test image determines the class of the latter.

In our experiments we substituted the k-means based clustering module with the adaptive mean shift (AMS) based robust nonparametric clustering. Thus, the textons instead of being *mean based* are now *mode based*, and the number of the significant ones is determined automatically.

The complete Brodatz database containing 112 textures with varying degrees of complexity was used in the experiments. Classification of the Brodatz database is challenging because it contains many nonhomogeneous textures. The 512×512 images were divided into four 256×256 subimages with half of the subimages being used for training (224 models) and the other half for testing (224 queries). The normalizations recommended in [18] (both in the image and filter domains) were also performed.

The number of significant textons detected with the AMS procedure depends on the texture. We have limited the number of mode textons extracted from a texture class to five. The same number was used for the mean textons. Thus, by adding the textons to the library, a texton histogram has at most $T = 560$ bins.

Table 2: Classification Results for the Brodatz Database

Filter	M4	M8	S	LM
RND	84.82%	88.39%	89.73%	92.41%
k-means	85.71%	94.64%	93.30%	97.32%
AMS	85.27%	93.75%	93.30%	98.66%

The classification results using the different filter banks are presented in Table 2. The best result was obtained with the LM mode textons, an additional three correct classifications out of the six errors with the mean textons. However, there is no clear advantage in using the mode textons with the other filter banks.

The classification performance is close to its upper bound defined by the texture inhomogeneity, due to which the test and training images of a class can be very different. This observation is supported by the performance degradation obtained when the database images were divided into sixteen 128×128 subimages and the same half/half partition yielded 896 models and 896 queries. The recognition rate decreased for all the filter banks. The best result of 94%, was again obtained with the LM filters for both the mean and mode textons. In [8], with the same setup but employing a different texture representation, and using only 109 textures from the Brodatz database the recognition rate was 80.4%.

A texture class is characterized by the histogram of the textons, an approximation of the feature space distribution. The histogram is constructed from a Voronoi diagram with T cells. The vertices of the diagram are the textons, and each histogram bin contains the number of feature points in a cell. Thus, variations in textons translate in approximating the distribution by a different diagram, but it appears to have a weak influence on the classification performance. When by uniform sampling five random vectors were chosen as textons, the classification performance (RND) decreased only between 1% to 6%.

The k-means clustering imposes rigidly a given number of identical spherical clusters over the feature space. Thus, it is expected that when this structure is not adequate, the mode based textons will provide a more meaningful decomposition of the texture image. This is proven in the following two examples.

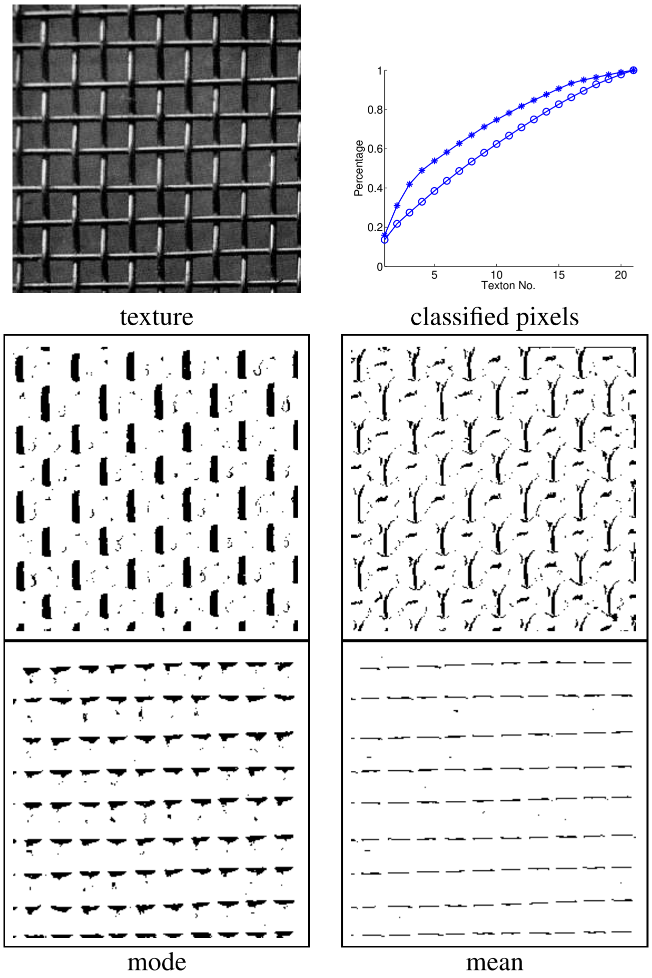


Figure 5: Mode ($*$) vs. mean ($\circ$) based textons. The local structure is better captured by the mode textons. D001 texture, LM filter bank.

In Figure 5 the LM filter bank was applied to a regular texture. The AMS procedure extracted 21 textons, the number also used in the k-means clustering. However, when ordered by size, the first few mode textons are associated with more pixels in the image than the mean textons, which always account for a similar number of pixels. The difference between the mode and mean textons can be seen by marking the pixels associated with textons of the same local structure (Figure 5, bottom). The advantage of the mode based representation is more evident for the nonregular texture in Figure 6, where the cumulative distribution of the mode textons classified pixels is has a sharper increase.

Since textons capture local spatial configurations, we believe that combining the mode textons with the representation proposed in [19] can offer more insight into why the texton approach is superior to previous techniques.

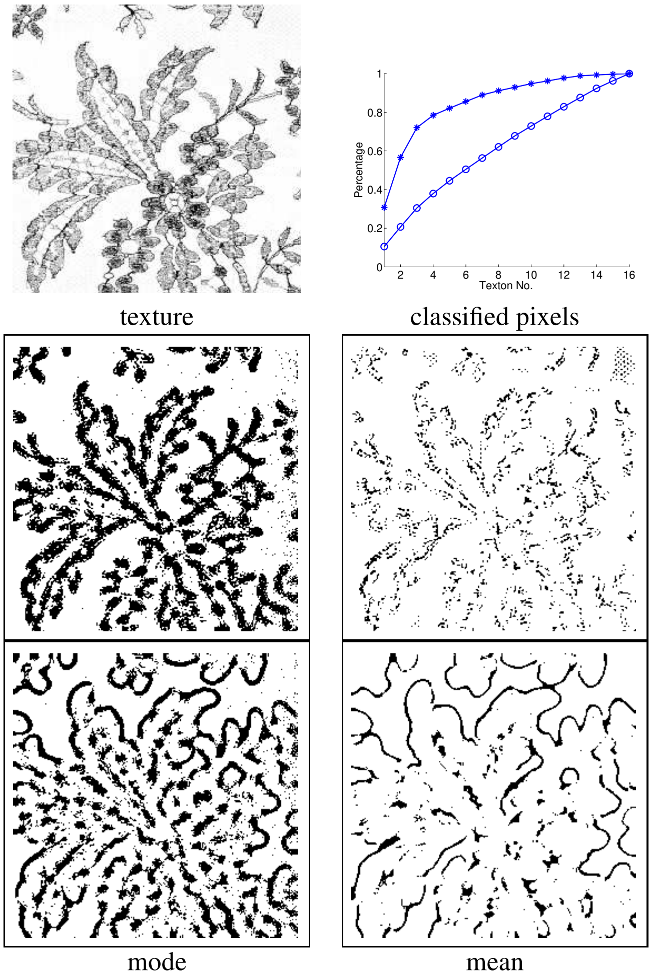


Figure 6: Mode ($*$) vs. mean ($\circ$) based textons. The local structure is better captured by the mode textons. D040 texture, S filter bank.

6. Conclusion

We have introduced a computationally efficient method that makes possible in high dimensional spaces the detection of the modes of distributions. By employing a data structure based on locality-sensitive hashing, a significant decrease in the running time was obtained while maintaining the quality of the results. The new implementation of the mean shift procedure opens the door to the development of vision algorithms exploiting feature space analysis – including learning techniques – in high dimensions. The C++ source code of this implememtation of mean shift can be downloaded from

`http://www.caip.rutgers.edu/riul`

Acknowledgments

We thank Bogdan Matei of the Sarnoff Corporation, Princeton, NJ, for calling our attention to the LSH data structure. This work was done during the sabbatical of I.S. at Rutgers University. The support of the National Science Foundation under the grant IRI 99-87695 is gratefully acknowledged.

References

[1] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(5):603–619, 2002.

[2] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. 8th Intl. Conf. on Computer Vision,* Vancouver, Canada, volume I, pages 438–445, July 2001.

[3] O. G. Cula and K. J. Dana. Compact representation of bidirectional texture functions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, Hawaii, volume 1, pages 1041–1047, 2001.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwartzkopf. *Computational Geometry. Algorithms and Applications*. Springer, second edition, 1998.

[5] D. DeMenthon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Proc. Statistical Methods in Video Processing Workshop,* Copenhagen, Denmark, 2002. Also CAR-TR-978 Center for Automat. Res., U. of Md, College Park.

[6] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, second edition, 2001.

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Int. Conf. on Very Large Data Bases*, pages 518–529, 1999.

[8] G. M. Haley and B. S. Manjunath. Rotation-invariant texture classification using a complete space-frequency model. *IEEE Trans. Image Process.*, 8(2):255–269, 1999.

[9] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 154–159, 1999.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. Symp. on Theory of Computing*, pages 604–613, 1998.

[11] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. Symp. on Theory of Computing*, pages 614–623, 1998.

[12] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Intl. J. of Computer Vision*, 43(1):29–44, 2001.

[13] S.A. Nene and S.K. Nayar. A simple algorithm for nearest-neighbor search in high dimensions. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(9):989–1003, 1997.

[14] R. Ostrovsky and Y. Rabani. Polynomial time approximation schemes for geometric k-clustering. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 349–358, 2000.

[15] C. Schmid. Constructing models for content-based image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, Hawaii, volume 2, pages 39–45, 2001.

[16] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc. 9th Intl. Conf. on Computer Vision,* Nice, France, 2003.

[17] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[18] M. Varma and A. Zisserman. Classifying images of materials. In *Proc. European Conf. on Computer Vision,* Copenhagen, Denmark, volume III, pages 255–271, 2002.

[19] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Madison, Wisconsin, volume II, pages 691–698, 2003.