Quick Shift and Kernel Methods for Mode Seeking

Andrea Vedaldi and Stefano Soatto

University of California, Los Angeles
Computer Science Department
{vedaldi,soatto}@ucla.edu

Abstract. We show that the complexity of the recently introduced medoid-shift algorithm in clustering N points is $O(N^2)$, with a small constant, if the underlying distance is Euclidean. This makes medoid shift considerably faster than mean shift, contrarily to what previously believed. We then exploit kernel methods to extend both mean shift and the improved medoid shift to a large family of distances, with complexity bounded by the effective rank of the resulting kernel matrix, and with explicit regularization constraints. Finally, we show that, under certain conditions, medoid shift fails to cluster data points belonging to the same mode, resulting in over-fragmentation. We propose remedies for this problem, by introducing a novel, simple and extremely efficient clustering algorithm, called quick shift, that explicitly trades off under- and over-fragmentation. Like medoid shift, quick shift operates in non-Euclidean spaces in a straightforward manner. We also show that the accelerated medoid shift can be used to initialize mean shift for increased efficiency. We illustrate our algorithms to clustering data on manifolds, image segmentation, and the automatic discovery of visual categories.

1 Introduction

Mean shift [9, 3, 5] is a popular non-parametric clustering algorithm based on the idea of associating each data point to a mode of the underlying probability density function. This simple criterion has appealing advantages compared to other traditional clustering techniques: The structure of the clusters may be rather arbitrary and the number of clusters does not need to be known in advance.

Mean shift is not the only "mode seeking" clustering algorithm. Other examples include earlier graph-based methods [13] and, more recently, *medoid shift* [20]. Unlike mean shift, medoid shift extends easily to general metric spaces (i.e. spaces endowed with a distance). In fact, mean shift is essentially a gradient ascent algorithm [3, 5, 24] and the gradient may not be defined unless the data space has additional structure (e.g. Hilbert space or smooth manifold structure). While there have been recent efforts to generalize mean shift to non-linear manifolds [21], medoid shift does not require any additional steps to be used on curved spaces. Moreover, the algorithm is non-iterative and there is no need for a stopping heuristic. Its biggest disadvantage is its computational complexity [20]. Depending on the implementation, medoid shift requires between $O(dN^2 + N^3)$

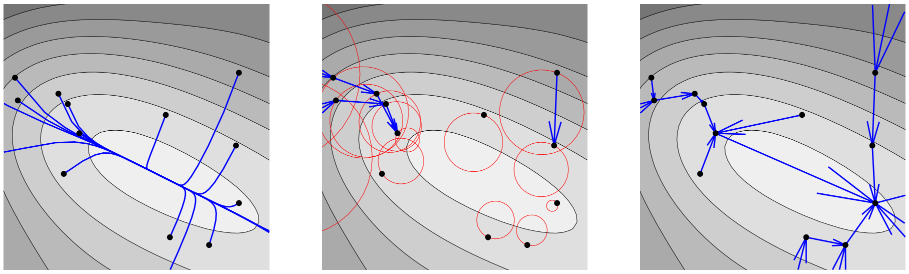


Fig. 1. Mode seeking algorithms. Comparison of different mode seeking algorithms (Sect. 2) on a toy problem. The black dots represent (some of) the data points $x_i \in \mathcal{X} \subset \mathbb{R}^2$ and the intensity of the image is proportional to the Parzen density estimate $P(x)$. **Left.** Mean shift moves the points uphill towards the mode approximately following the gradient. **Middle.** Medoid shift approximates mean shift trajectories by connecting data points. For reason explained in the text and in Fig. 2, medoid shifts are constrained to connect points comprised in the red circles. This disconnects portions of the space where the data is sparse, and can be alleviated (but not solved) by iterating the procedure (Fig. 2). **Right.** Quick shift (Sect. 3) seeks the energy modes by connecting nearest neighbors at higher energy levels, trading-off mode over- and under-fragmentation.

and $O(dN^2 + N^{2.38})$ operations to cluster N points, where d is the dimensionality of the data. On the other hand, mean shift is only $O(dN^2T)$, where T is the number of iterations of the algorithm, and clever implementations yield $dT \ll N$.

Contributions. In this paper we show that the computational complexity of Euclidean medoid shift is only $O(dN^2)$ (with a small constant), which makes it faster (not slower!) than mean shift (Sect. 3). We then generalize this result to a large family of non-Euclidean distances by using kernel methods [18], showing that in this case the complexity is bounded by the effective dimensionality of the kernel space (Sect. 3). Working with kernels has other advantages: First, it extends to mean shift (Sect. 4); second, it gives an explicit interpretation of non-Euclidean medoid shift; third, it suggests why such generalized mode seeking algorithms skirt the *curse of dimensionality*, despite estimating a density in complex spaces (Sect. 4). In summary, we show that kernels extend mode seeking algorithms to non-Euclidean spaces in a simple, general and efficient way.

Can we conclude that medoid shift should replace mean shift? Unfortunately, not. We show that the weak point of medoid shift is its inability to identify consistently all the modes of the density (Sect. 2). This fact was addressed implicitly by [20] who reiterate medoid shift on a simplified dataset (similar to [2]). However, this compromises the non-iterative nature of medoid shift and changes the underlying density function (which may be undesirable). Moreover, we show that this fix does not always work (Fig. 2).

We address this issue in two ways. First, we propose using medoid shift to simplify the data and initialize the more accurate mean shift algorithm (Sect. 5.2 and Sect. 5.3). Second, we propose an alternative mode seeking algorithm that can trade off mode over- and under-fragmentation (Sect. 3). This algorithm, related to [13], is particularly simple and fast, yields surprisingly good segmentations, and returns a one parameter family of segmentations where model selection can be applied.

We demonstrate these algorithms on three tasks (Sect. 5): Clustering on a manifold (Sect. 5.1), image segmentation (Sect. 5.2), and clustering image signatures for automatic object categorization (Sect. 5.3). The relative advantages and disadvantages of the various algorithms are discussed.

2 Mode Seeking

Given N data points $x_1, \dots, x_N \in \mathcal{X} = \mathbb{R}^d$, a *mode seeking* clustering algorithm conceptually starts by computing the *Parzen density estimate*

$$P(x) = \frac{1}{N} \sum_{i=1}^{N} k(x - x_i), \quad x \in \mathbb{R}^d \tag{1}$$

where $k(x)$ can be a Gaussian or other window.[1] Then each point x_i is moved towards a mode of $P(x)$ evolving the trajectory $y_i(t)$, $t > 0$ uphill, starting from $y_i(0) = x_i$ and following the gradient $\nabla P(y_i(t))$. All the points that converge to the same mode form a cluster.

A mode seeking algorithm needs (i) a numerical scheme to evolve the trajectories $y_i(t)$, (ii) a halting rule to decide when to stop the evolution and (iii) a clustering rule to merge the trajectory end-points. Next, we discuss two algorithms of this family.

Mean Shift. Mean shift [9, 5] is based on an efficient rule to evolve the trajectories $y_i(t)$ when the window $k(x)$ can be written as $\psi(\|x\|_2^2)$ for a convex function $\psi(z)$ (for instance the Gaussian window has $\psi(z) \propto \exp(-z)$). The idea is to bound the window from below by the quadric $k(z') \geq k(z) + (\|z'\|_2^2 - \|z\|_2^2)\dot{\psi}(\|z\|_2^2)$. Substituting in (1) yields

$$P(y') \geq P(y) + \frac{1}{N} \sum_{j=1}^{N} (\|y' - x_j\|_2^2 - \|y - x_j\|_2^2)\dot{\psi}(\|y - x_j\|_2^2), \tag{2}$$

and maximizing this lower bound at $y = y_i(t)$ yields the mean-shift update rule

$$y_i(t+1) = \operatorname*{argmax}_y \frac{1}{N} \sum_{j=1}^{N} \|y - x_j\|_2^2 \dot{\psi}(\|y_i(t) - x_j\|_2^2) = \frac{\sum_{j=1}^{N} \dot{\psi}(\|y_i(t) - x_j\|_2^2) x_j}{\sum_{j=1}^{N} \dot{\psi}(\|y_i(t) - x_j\|_2^2)}. \tag{3}$$

[1] The term "kernel" is also used in the literature. Here we use the term "window" to avoid confusion with the kernels introduced in Sect. 3.

If the profile $\psi(z)$ is monotonically decreasing, then $P(y_i(t)) < P(y_i(t+1))$ at each step and the algorithm converges in the limit (since P is bounded [5]). The complexity is $O(dN^2T)$, where d is the dimensionality of the data space and T is the number of iterations. The behavior of the algorithm is illustrated in Fig. 1.

Medoid Shift. Medoid shift [20] is a modification of mean shift in which the trajectories $y_i(t)$ are constrained to pass through the points x_i, $i = 1, \dots, N$. The advantage of medoid shift are: (i) only one step $y_i(1)$, $i = 1, \dots, N$ has to be computed for each point x_i (because $y_i(t+1) = y_{y_i(t)}(1)$), (ii) there is no need for a stopping/merging heuristic (as these conditions are met exactly), and (iii) the data space $\mathcal{X}$ may be non-Euclidean (since to maximize (4) there is no need to compute derivatives). Eventually, points are linked by steps into a forest, with clusters corresponding to trees. The algorithm is illustrated in Fig. 1.

According to [20], the main drawback of medoid shift is speed. In fact, maximizing (3) restricted to the dataset amounts to calculating

$$y_i(1) = \operatorname*{argmax}_{y \in \{x_1, \dots, x_N\}} \frac{1}{N} \sum_{j=1}^{N} \mathsf{d}^2(y, x_j) \dot{\phi}(\mathsf{d}^2(x_j, x_i)) \tag{4}$$

where $\mathsf{d}^2(x,y) = \|x-y\|_2^2$ in the Euclidean case. A basic implementation requires $O(N^3 + dN^2)$ operations, assuming $O(d)$ operations to evaluate $\mathsf{d}^2(x,y)$. However, by defining matrices $D_{kj} = \mathsf{d}^2(x_k, x_j)$ and $F_{ki} = \dot{\phi}(D_{ik})/N$, we can rewrite (4) as

$$y_i(1) = \operatorname*{argmax}_{k=1,\dots,N} \sum_{j=1}^{N} D_{kj} F_{ji} = \operatorname*{argmax}_{k=1,\dots,N} e_k^\top D F e_i \tag{5}$$

where e_i denotes the i-th element of the canonical basis.[2] As noted in [20], $O(N^{2.38})$ operations are sufficient by using the fastest matrix multiplication algorithm available. Unfortunately the hidden constant of this algorithm is too large to be practical (see [12], pag. 501). Thus, a realistic estimate of the time required is more pessimistic than what suggested by the asymptotic estimate $O(dN^2 + N^{2.38})$.

Here we note that a more delicate issue with medoid shift is that it may fail to properly identify the modes of the density $P(x)$. This is illustrated in Fig. 2, where medoid shift fails to cluster three real points $-1, +1$ and $+1/2$, finding two modes -1 and $+1$ instead of one. To overcome this problem, [20] applies medoid shift iteratively on the modes (in the example -1 and $+1$). However, this solution is not completely satisfactory because (i) the underlying model $P(x)$ is changed (similarly to *blurry* mean shift [9, 3]) and (ii) the strategy does not work in all cases (for instance, in Fig. 2 points -1 and $+1$ still fail to converge to a single mode).

Finally, consider the interpretation of medoid shift. When $\mathcal{X}$ is a Hilbert space, medoid (and mean) shift follow approximately the gradient of the density

[2] For instance $e_2 = \begin{bmatrix} 0 \ 1 \ 0 \dots 0 \end{bmatrix}^\top$.

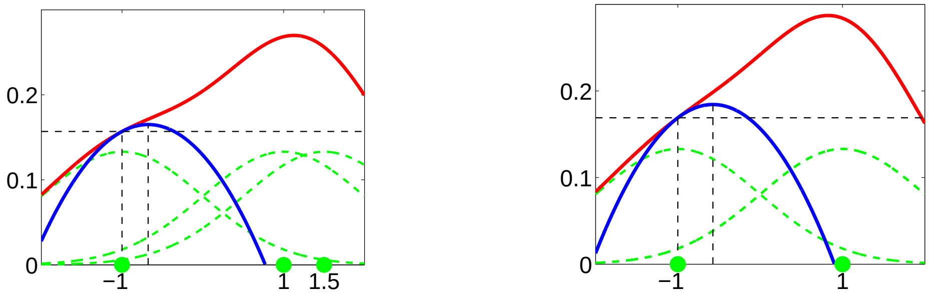


Fig. 2. Medoid shift over-fragmentation. Left. We apply medoid shift to cluster points $-1, +1, +1/2 \in \mathbb{R}$ using a Gaussian window of variance $\sigma^2 = 1$ (dashed green lines). The density $P(x)$ (red curve; Sect. 2) has a single mode, but medoid shift fails to move the point -1 towards the mode (i.e. $y_{-1}(1) = -1$). The reason is that the quadratic lower bound (2) (blue curve) is larger at -1 than it is at $+1$ or $+1/2$. Notice that mean shift would have moved -1 towards the mode by a small, but finite amount, eventually extracting the single mode. **Right.** The problem is not solved even if medoid shift is reiterated [20] on the two modes -1 and $+1$ (where $+1$ has double mass), even if the density $P(x)$ *does* become blurrier [2, 20].

$P(x)$ (by maximizing the lower bound (3)). The gradient depends crucially on the inner product and corresponding metric defined on $\mathcal{X}$, which encodes the cost of moving along each direction [22]. For general metric spaces $\mathcal{X}$, the gradient may not be defined, but the term $\mathsf{d}^2(x, y)$ in (4) has a similar direction-weighing effect. In later sections we will make this connection more explicit.

3 Fast Clustering

Faster Euclidean Medoid Shift. We show that the complexity of Euclidean medoid shift is only $O(dN^2)$ (with a small constant) instead of $O(dN^2 + N^{2.38})$ (with a large constant) [20]. Let $X = \begin{bmatrix} x_1 \dots x_N \end{bmatrix}$ be the data matrix. Let $n = (X^\top \odot X^\top)\mathbf{1}$ be the vector of the squared norms of the data, where $\mathbf{1}$ denotes the vector of all ones and $\odot$ the Hadamard (component wise) matrix product. Then we have

$$D = \mathbf{1}n^\top + n\mathbf{1}^\top - 2X^\top X, \qquad DF = n(\mathbf{1}^\top F) + \mathbf{1}(n^\top F) - 2X^\top (XF). \tag{6}$$

The term $\mathbf{1}(n^\top F)$ has constant columns and is irrelevant to the maximization (5). Therefore, we need to compute

$$DF \propto n(\mathbf{1}^\top F) - 2X^\top (XF), \quad n = (X^\top \odot X^\top)\mathbf{1} = (I \odot X^\top X)\mathbf{1} \tag{7}$$

where I is the identity matrix.[3] It is now easy to check that each matrix product in (7) requires $O(dN^2)$ operations only.

[3] And we used the fact that $(I \odot AB)\mathbf{1} = (B^\top \odot A)\mathbf{1}$.

Kernel Medoid Shift. An advantage of medoid shift is the possibility of computing (4) for distances $\mathsf{d}^2(x,y)$ other than the Euclidean one [20]. The decomposition (6) can still be carried out if the distance $\mathsf{d}^2(x,y)$ can be expressed as $K(x,x)+K(y,y)-2K(x,y)$ for an appropriate positive definite (p.d.) kernel[4] K [18]. Then we have $D = \mathbf{1}n^\top + n\mathbf{1}^\top - 2K$, and

$$DF \propto n(\mathbf{1}^\top F) - 2KF, \quad n = (I \odot K)\mathbf{1}.$$

Unfortunately, the multiplication KF is still $O(N^{2.38})$. However, we can search for a low-rank decomposition $G^\top G$ of K (we assume, without loss of generality, that K is centered[5]). If G is a decomposition of rank d, then

$$DF \propto n(\mathbf{1}^\top F) - 2G(G^\top F), \quad n = (I \odot G^\top G)\mathbf{1} = (G^\top \odot G^\top)\mathbf{1}$$

can still be computed in $O(dN^2)$ operations. The cost of decomposing K is typically around $O(d^2N)$ [8, 1]. See Fig. 3 for a basic implementation.

Quick Shift. In order to seek the mode of the density $P(x)$, it is not necessary to use the gradient or the quadratic lower bound (2). Here we propose *quick shift*, which simply moves each point x_i to the nearest neighbor for which there is an increment of the density $P(x)$. In formulas,

$$y_i(1) = \operatorname*{argmin}_{j: P_j > P_i} D_{ij}, \quad P_i = \frac{1}{N}\sum_{j=1}^{N} \phi(D_{ij}). \tag{8}$$

Quick shift has four advantages: (i) simplicity; (ii) speed ($O(dN^2)$ with a small constant); (iii) generality (the nature of D is irrelevant); (iv) a tuning parameter to trade off under- and over-fragmentation of the modes. The latter is obtained because there is no *a-priori* upper bound on the length D_{ij} of the shifts $y_i(0) \to y_i(1)$. In fact, the algorithm connects all the points into a single tree. Modes are then recovered by breaking the branches of the tree that are longer than a threshold τ. Searching τ amounts to performing model selection and balances under- and over-fragmentation of the modes. The algorithm is illustrated in Fig. 1.

Quick shift is related to the classic algorithm from [13]. In fact, we can rewrite (8) as

$$y_i(1) = \operatorname*{argmax}_{j=1,\dots,N} \frac{\operatorname{sign}(P_j - P_i)}{D_{ij}}, \quad \text{and compare it to } y_i(1) = \operatorname*{argmax}_{j: d(x_j, x_i) < \tau} \frac{P_j - P_i}{D_{ij}} \tag{9}$$

[4] The kernel K should not be confused with the Parzen window $k(z)$ appearing in (1). In the literature, it is common to refer to the Parzen window as "kernel", but in most cases it has rather different mathematical properties than the kernel K we consider here. An exception is when the window is Gaussian, in which cases $k(\mathsf{d}^2(x,y))$ is a p.d. kernel. In this case, we point out an interesting interpretation of mean shift as a local optimization algorithm that, starting from each data point, searches for the pre-image of the global data average computed in kernel space. This explains the striking similarity of the mean shift update Eq. (3) and Eq. (18.22) of [19].

[5] K is *centered* if $K\mathbf{1} = 0$. If this is not the case, we can replace K by $K' = HKH$, where $H = I - \mathbf{1}\mathbf{1}^\top/N$ is the so-called *centering matrix*. This operation translates the origin of the kernel space, but does not change the corresponding distance.

as given by [13]. Notice that $(P_j - P_i)/D_{ij}$ is a numerical approximation of the gradient of P in the direction $x_j - x_i$. The crucial difference is that maximizing the gradient approximation must be done in a neighborhood of each point defined *a-priori* by the choice of the parameter τ. Thus, model selection in [13] requires running the algorithm multiple times, one for each value of τ. In contrast, quick shift returns at once the solutions for all possible values of τ, making model selection much more efficient.

4 Cluster Refinement

In the previous section we introduced fast kernel medoid shift as an accelerated version of non-Euclidean medoid shift. Since medoid shift may over-fragment modes, quick shift was then proposed as a method to control under- and over-fragmentation by the choice of a parameter τ. No algorithm, however, guarantees the same accuracy of the slower mean shift.

It is then natural to ask whether mean shift could be extended to work in a non-Euclidean setting. [20] cites the problem of defining the mean as the major obstacle to this idea. [21] addresses this issue by defining mean shift vectors on the tangent space of a non-linear manifold, but no proof of convergence is given, and the applicability is limited by the fact that the data space needs to have a manifold structure known analytically.

A simple solution to this problem is to extend kernel medoid to a corresponding kernel mean shift procedure. Let $K(\cdot,\cdot)$ be a p.d. kernel on the data space $\mathcal{X}$. Then $K(x,\cdot)$ is an element of the so called *reproducing kernel Hilbert space* $\mathcal{H}$ [19], whose inner product is defined by letting $\langle K(x,\cdot), K(y,\cdot)\rangle_{\mathcal{H}} = K(x,y)$. Points $x \in \mathbb{R}^d$ are then identified with elements $K(x,\cdot)$ of the Hilbert space. Given this identification, we can write $\langle \cdot, x\rangle_{\mathcal{H}}$ for $\langle \cdot, K(x,\cdot)\rangle_{\mathcal{H}}$.

Kernel mean shift computes a "density[6]" on $\mathcal{H}$

$$P(y) = \frac{1}{N}\sum_{j=1}^{N} k(\mathsf{d}^2_{\mathcal{H}}(y, x_j)), \qquad y \in \mathcal{H} \tag{10}$$

where $\mathsf{d}^2_{\mathcal{H}}(x_j, y) = \langle y, y\rangle_{\mathcal{H}} + \langle x_j, x_j\rangle_{\mathcal{H}} - 2\langle y, x_j\rangle_{\mathcal{H}}$. Notice that $y \in \mathcal{H}$, unlike standard mean shift, does not belong necessarily to the data space $\mathcal{X}$ (up to the identification $x \equiv K(x,\cdot)$). However, if $k(z)$ is monotonically decreasing, then maximizing w.r.t. y can be restricted to the linear subspace $\operatorname{span}_{\mathcal{H}} X = \operatorname{span}_{\mathcal{H}}\{x_1, \dots, x_n\} \subset \mathcal{H}$ (if not, the orthogonal projection of y onto that space decreases simultaneously all terms $\mathsf{d}^2_{\mathcal{H}}(x_j, y)$).

Therefore, we can express all calculations relative to $\operatorname{span}_{\mathcal{H}} X$. In particular, if $K_{ij} = K(x_i, x_j)$ is the kernel matrix, we have $\mathsf{d}^2_{\mathcal{H}}(x_j, y) = y^\top K y + e_j^\top K e_j - 2 e_j^\top K y$ where e_j is the j-th vector of the canonical basis and y is a vector of N coefficients. As in standard mean shift, the shifts are obtained by maximizing

[6] The interpretation is discussed later.

(Kernel) Mean Shift

```
function Z = meanshift(G, sigma)

[d,N]  = size(G) ;
oN     = ones(N,1) ;
od     = ones(d,1) ;
n      = (G'.*G')*od ;

Z = G ;
T = 100 ;
for t=1:T
  m = (Z'.*Z')*od ;
  D = m*oN' + oN*n' - 2*(Z'*G) ;
  F = - exp(- .5 * D' / sigma^2) ;
  Y = F ./ (oN * (oN'*F)) ;
  Z = G*Y ;
end
```

(Kernel) Medoid Shift

```
function map = medoidshift(G, sigma)

[d,N]  = size(G) ;
oN     = ones(N,1) ;
od     = ones(d,1) ;
n      = (G'.*G')*od ;

D = n*oN' + oN*n' - 2*(G'*G) ;
F = - exp(- .5 * D' / sigma^2) ;
Q = n * (oN'*F) - 2 * G' * (G*F) ;

[drop,map]  = max(Q) ;
```

Fig. 3. Kernel mean and medoid shift algorithms. We show basic MATLAB implementations of two of the proposed algorithms. Here $K = G^\top G$ is a low-rank decomposition $G \in \mathbb{R}^{d\times N}$ of the (centered) kernel matrix and `sigma` is the (isotropic) standard deviation of the Gaussian Parzen window. Both algorithms are $O(dN^2)$ (for a fixed number of iterations of mean shift), reduce to their Euclidean equivalents by setting $G \equiv X$ and $Z \equiv Y$, and can be easily modified to use the full kernel matrix K rather than a decomposition $G^\top G$ (but the complexity grows to $O(N^3)$).

the lower bound

$$y_i(t+1) = \operatorname*{argmax}_{y\in\mathbb{R}^N} \sum_{j=1}^{N} (y^\top K y + e_j^\top K e_j - 2e_j^\top K y)\dot{\phi}(\mathsf{d}^2_{\mathcal{H}}(x_j, y_i(t))).$$

Deriving w.r.t. y and setting to zero yields the update equation

$$y_i(t+1) = \frac{1}{\mathbf{1}^\top F e_i}(F e_i), \quad F_{ji} = \dot{\phi}(D_{ij}), \quad D_{ij} = \mathsf{d}^2_{\mathcal{H}}(y_i(t), x_j). \tag{11}$$

Low-rank approximation. Similarly to medoid shift, we can accelerate the algorithm by using a low-rank decomposition $K = G^\top G$ of the (centered) kernel matrix. It is useful to switch to matrix notation for all the quantities. Let $Y = \begin{bmatrix} y_1, \dots\ y_M \end{bmatrix}$ be the trajectory matrix and define $Z = GY$ the reduced coordinates.[7] The distance matrix D can be written compactly as

$$D = m\mathbf{1}^\top + \mathbf{1}n^\top - 2Y^\top K = m\mathbf{1}^\top + \mathbf{1}n^\top - 2Z^\top G;$$

where

$$m = (Y^\top \odot Y^\top K)\mathbf{1} = (Z^\top \odot Z^\top)\mathbf{1}, \quad n = (I \odot K)\mathbf{1} = (G^\top \odot G^\top)\mathbf{1}.$$

At each iteration D is calculated in $O(dN^2)$ operations. Then $F = \dot{\phi}(D^\top)/N$ is evaluated component-wise. Finally the trajectories Y (or equivalently Z) are updated by

$$Y \leftarrow F\,\mathrm{diag}(F\mathbf{1})^{-1}, \qquad Z \leftarrow GY.$$

[7] Similarly, the data matrix X has reduced coordinates equal to G.

in $O(N^2)$ operations. Notice that, by setting $G \equiv X$ and $Z \equiv Y$ in these equations, we obtain Euclidean mean shift back. See Fig. 3 for a basic implementation.

Interpretation, Regularization and Scaling. In Euclidean mean shift the function $P(x)$ is a non-parametric estimate of a probability density. Does the same interpretation hold in kernel space? For any fixed data set of size N, we can restrict our attention to the subspace space $\operatorname{span}_{\mathcal{H}} X \subset \mathcal{H}$ and interpret $P(x)$ as a probability density on this finite-dimensional space. Unfortunately, the number of dimensions of this space may be as large as the number of data points N, which makes the Parzen density estimate $P(x)$ inconsistent (in the sense that the variance does not converge to zero as $N \to \infty$). So how do we make sense of kernel mean shift? The idea is to use the fact that most of the dimensions of $\operatorname{span}_{\mathcal{H}} X$ are often unimportant. Formally, consider the eigen-decomposition $K = V\Sigma V^\top = G^\top G$, $G = \Sigma^{\frac{1}{2}} V^\top$ of the (centered) kernel matrix K. Assume $\Sigma^{\frac{1}{2}} = N \operatorname{diag}(\sigma_1, \dots, \sigma_N)$, with $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N$. According to this decomposition, vectors $x, y \in \operatorname{span}_{\mathcal{H}} X$ can be identified with their coordinates $g, z \in \mathbb{R}^N$ so that $\langle x, y \rangle_{\mathcal{H}} = \langle g, z \rangle$. Moreover the data matrix $G = \begin{bmatrix} g_1 \dots g_n \end{bmatrix}$ has null mean[8] and covariance $GG^\top / N = \Sigma / N = \sigma_1^2 \operatorname{diag}(\lambda_1^2, \dots, \lambda_N^2)$. If λ_i decay fast, the effective dimension of the data can be much smaller than N.

The simplest way to regularize the Parzen estimate is therefore to discard the dimensions above some index d (which also improves efficiency). Another option is to blur the coordinates z by adding a small Gaussian noise η of isotropic standard deviation ϵ, obtaining a regularized variable $z' = z + \eta$. The components of z with smaller variance are "washed out" by the noise, and we can obtain a consistent estimator of z' by using the regularized Parzen estimate $\sum_{i=1}^N (g_\epsilon * k)(z_i)$ (the same idea is implicitly used, for instance, in kernel Fisher discriminant analysis [15], where the covariance matrix computed in kernel space is regularized by the addition of $\epsilon^2 I$). This suggests that using a kernel with sufficient isotropic smoothing may be sufficient.

Finally, we note that, due to the different scalings $\lambda_1, \dots, \lambda_N$ of the linear dimensions, it might be preferable to use an adapted Parzen window, which retains the same proportions [17]. This, combined with the regularization ϵ, suggests us to scale each axis of the kernel by $\sqrt{\sigma^2 \lambda_i^2 + \epsilon^2}$.[9]

[8] Because K is assumed to be centered, so that $\mathbf{1}^\top G^\top (G\mathbf{1}) = \mathbf{1}^\top K \mathbf{1} = 0$.

[9] So far we disregarded the normalization constant of the Parzen window $k(x)$ as it was irrelevant for our purposes. If, however, windows $k_\sigma(x)$ of variable width σ are used [6], then the relative weights of the windows become important. Recall that in the d dimensional Euclidean case one has $k_\sigma(0)/k_{\sigma'}(0) = (\sigma'/\sigma)^d$. In kernel space therefore one would have

$$\frac{k_\sigma(0)}{k_{\sigma'}(0)} = \sqrt{\prod_{i=1}^N \frac{\sigma'^2 \lambda_i^2 + \epsilon^2}{\sigma^2 \lambda_i^2 + \epsilon^2}}.$$

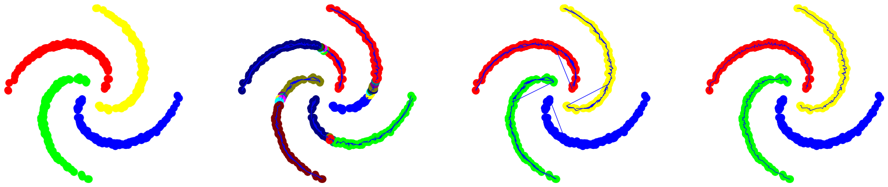


Fig. 4. Clustering on a manifold. By using kernel ISOMAP we can apply kernel mean and medoid shift to cluster points on a manifold. For the sake of illustration, we reproduce an example from [20]. From left to right: Kernel mean shift (7.8s), *non-iterated* kernel medoid shift (0.18s), iterated kernel medoid shift (0.48s), quick shift (0.12s). We project the kernel space to three dimensions $d = 3$ as the residual dimensions are irrelevant. All algorithms but non-iterated medoid shift segment the modes successfully. Compared to [20], medoid shift has complexity $O(dN^2)$, (with a small constant and $d = 3 \ll N$) instead of $O(N^3)$ (small constant) or $O(N^{2.38})$ (large constant)

5 Applications

5.1 Clustering on Manifolds

[20] applies medoid shift to cluster data on manifolds, based on the distance matrix D calculated by ISOMAP. If the kernel matrix $K = HDH'/2$, $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ is p.d., we can apply directly kernel mean or medoid shift to the same problem. If not, we can use the technique from [4] to regularize the estimate and enforce this property. In Fig. 4 this idea is used to compare kernel mean shift, kernel medoid shift and quick shift in a simple test case.

5.2 Image Segmentation

Image segmentation is a typical test case for mode seeking algorithms [5, 16, 20]. Usually mode seeking is applied to this task by clustering data $\{(p, f(p)), p \in \Omega\}$, where $p \in \Omega$ are the image pixels and $f(p)$ their color coordinates (we use the same color space of [5]).

As in [5], we apply mean shift to segment the image into super-pixels (mean shift variants can be used to obtain directly full segmentations [2, 16, 24]). We compare the speed and segmentation quality obtained by using mean shift, medoid shift, and quick shift (see Fig. 5 for further details).

Mean shift is equivalent to [5] and can be considered a reference to evaluate the other segmentations. Non-iterative medoid shift (first column) over-fragments significantly (see also Fig. 2), which in [20] is addressed by reiterating the algorithm. However, since our implementation is only $O(dN^2)$, medoid shift has at least the advantage of being much faster than mean shift, and can be used to speed up the latter. In Fig. 5 we compare the time required to run mean shift from scratch and from the modes found by medoid shift. We report the speedup (as the number of modes found by medoid shift over the number of pixels), the computation time of medoid+mean shift and, in brackets, the computation time

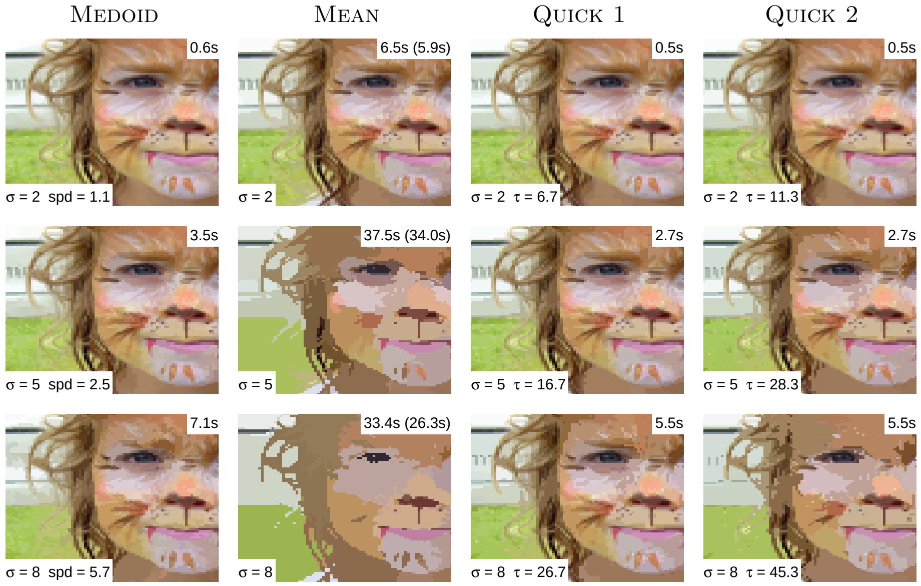


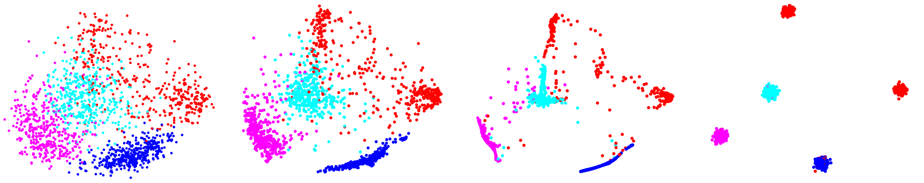
Fig. 5. Image segmentation. We compare different mode seeking techniques for segmenting an image (for clarity we show only a detail). We report the computation time in seconds (top-right corner of each figure). In order to better appreciate the intrinsic efficiency advantages of each method, we use comparable vanilla implementations of the algorithms (in practice, one could use heuristics and advanced approximation techniques [23] to significantly accelerate the computation). We use a Gaussian kernel of isotropic standard deviation σ in the spatial domain and use only one optimization: We approximate the support of the Gaussian window by a disk of radius 3σ (in the spatial domain) which results in a sparse matrix F. Therefore the computational effort increases with σ (top to bottom). The results are discussed in the text.

of the mean shift part only. Interestingly, the efficiency increases for larger σ, so that the overall computation time actually *decreases* when σ is large enough.

Finally, we show the result of quick shift segmentation (last two columns) for increasing values of the regularization parameter τ. Notice that quick shift is run only once to get both segmentations (Sect. 3) and that the algorithm is in practice much faster than the other two, while still producing reasonable super-pixels.

5.3 Clustering Bag-of-Features

The interesting work [11] introduces a large family of positive definite kernels for probability measures which includes many of the popular metrics: χ^2 kernel, Hellinger's kernel, Kullback-Leibler kernel and l^1 kernel. Leveraging on these ideas, we can use kernel mean shift to cluster *probability measures*, and in particular histograms, such as the ones arising in bag-of-features [7] or similar rep-



Algorithm	Scal.	Rate [%]	Time [s]
mean	no	93.9	29.9
mean	yes	94.6	89.0
mean+medoid	no	93.9	3.63
mean+medoid	yes	94.5	6.28
quick	no	96.3	0.62
quick	yes	96.1	0.70

Fig. 6. Automatic visual categorization. We use kernel mean shift to cluster bag-of-features image descriptors of 1600 images from Caltech-4 (four visual categories: airplanes, motorbikes, faces, cars). **Top.** From left to right, iterations of kernel mean shift on the bag-of-features signatures. We plot the first two dimensions of the rank-reduced kernel space (z vectors) and color the points based on the ground truth labels. In the rightmost panel the data converged to five points, but we artificially added random jitter to visualize the composition of the clusters. **Bottom.** Samples from the five clusters found (notice that airplane are divided in two categories). We also report the clustering quality, as the percentage of correct labels compared to the ground truth (we merge the two airplanes categories into one), and the execution time. We use basic implementations of the algorithms, although several optimizations are possible.

resentations. In the rest of the section we experiment with the χ^2 kernel

$$K_{\chi^2}(x,y) = 2\sum_{b=1}^{B} \frac{x_b y_b}{x_b + y_b}$$

where x and y are histograms of B bins.

Inspired by [10], we attempt to automatically infer the object categories of Caltech-4 in a completely unsupervised setting. We select at random 1600 images from the categories bike, airplanes, cars and faces. Instead of the more sophisticated representation of [10], we compute a basic bag-of-feature image representation as suggested by [25]: We extract multiscale Harris and DoG interest points (of fixed orientation; see [25] and ref. therein) and calculate SIFT descriptors [14], obtaining about 10^3 features per image. We then generate a vocabulary of 400 visual words by clustering a random selection of such descriptors by using k-means. For each image, we compute a bag-of-feature histogram x by counting the number of occurrences of each visual word in that image. Finally, we use the χ^2 kernel to generate the kernel matrix, that we feed to our clustering algorithms.

In Fig. 6 we compare kernel mean shift, kernel mean shift initialized by medoid shift, and quick shift. The problem we solve is considerably harder than [10], since in our case the number of clusters (categories) is unknown. All algorithms discover five (rather than four) categories (Fig. 6), but the result is quite reasonable since the category airplanes contains two distinct and visually quite different populations (grounded and airborne airplanes). Moreover, compared to [10] we do not try to explicitly separate an object from its background, but we use a simple holistic representation of each image.

The execution time of the algorithms (Fig. 6) is very different. Mean shift is relatively slow, at least in our simple implementation, and its speed greatly improves when we use medoid shift to initialize it. However, consistently with our image segmentation experiments, quick shift is much faster.

We also report the quality of the learned clusters (after manually merging the two airplane subcategories) as the percentage of correct labels. Our algorithm performs better than [10], that uses spectral clustering and reports 94% accuracy on selected prototypes and as low as 85% when all the data are considered; our accuracy in the latter case is at least 94%. We also study rescaling as proposed in Sect. 4, showing that it (marginally) improves the results of mean/medoid shift, but makes the convergence slower. Interestingly, however, the best performing algorithm (not to mention the fastest) is quick shift.

6 Conclusions

In this paper we exploited kernels to extend mean shift and other mode seeking algorithms to a non-Euclidean setting. This also clarifies issues of regularization and data scaling when complex spaces are considered. In this context, we showed how to derive a very efficient version of the recently introduced medoid shift algorithm, whose complexity is *lower* than mean shift. Unfortunately, we also showed that medoid shift often results in over-fragmented clusters. Therefore, we proposed to use medoid shift to initialize mean shift, yielding a clustering algorithm which is both efficient and accurate.

We also introduced quick shift, which can balance under- and over-fragmentation of the clusters by the choice of a real parameter. We showed that, in practice, this algorithm is very competitive, resulting in good (and sometimes better) segmentations compared to mean shift, at a fraction of the computation time.

Acknowledgment. Supported by AFOSR FA9550-06-1-0138 and ONR N00014-08-1-0414.

References

1. F. R. Bach and M. I. Jordan. Kernel independent componet analysis. *Journal of Machine Learninig Research*, 3(1), 2002.
2. M. Carreira-Perpiñán. Fast nonparametric clustering with gaussian blurring mean-shift. In *Proc. ICML*, 2006.
3. Y. Cheng. Mean shift, mode seeking, and clustering. *PAMI*, 17(8), 1995.
4. H. Choi and S. Choi. Robust kernel isomap. *Pattern Recognition*, 2006.
5. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5), 2002.
6. D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *Proc. ICCV*, 2001.
7. G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV*, 2004.
8. S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learninig Research*, 2001.
9. K. Fukunaga and L. D. Hostler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. on Information Theory*, 21(1), 1975.
10. K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *Proc. CVPR*, 2006.
11. M. Hein and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. In *Proc. AISTAT*, 2005.
12. D. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Third Edition, 1998.
13. W. L. G. Koontz, P. Narendra, and K. Fukunaga. A graph-theoretic approach to nonparametric cluster analyisis. *IEEE Trans. on Computers*, c-25(9), 1976.
14. D. Lowe. Implementation of the scale invariant feature transform. http://www.cs.ubc.ca/ lowe/keypoints/, 2007.
15. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In *Proc. IEEE Neural Networks for Signal Processing Workshop*, 1999.
16. S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *Proc. CVPR*, 2007.
17. S. R. Sain. Multivariate locally adaptive density estimation. *Comp. Stat. and Data Analysis*, 39, 2002.
18. B. Schölkopf. The kernel trick for distances. *Proc. NIPS*, 2001.
19. B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.
20. Y. A. Sheikh, E. A. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *Proc. CVPR*, 2007.
21. R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *Proc. CVPR*, 2006.
22. G. Sundaramoorthy, A. Yezzi, and A. Mennucci. Sobolev active contours. *Int. J. Comput. Vision*, 73(3), 2007.
23. C. Yang, R. Duraiswami, N. A. Gumerov, and L. Davis. Improved fast Gauss transform and efficient kernel density estimation. In *Proc. ICCV*, 2003.
24. X. Yuan and S. Z. Li. Half quadric analysis for mean shift: with extension to a sequential data mode-seeking method. In *Proc. CVPR*, 2007.
25. J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 2006.