Mode-Detection via Median-Shift

Lior Shapira
Tel-Aviv University
Tel Aviv, Israel
liors@post.tau.ac.il

Shai Avidan
Adobe Systems, Inc.
275 Grove St. Newton, MA
avidan@adobe.com

Ariel Shamir
The Interdisciplinary Center
Herzliya, Israel
arik@idc.ac.il

Abstract

Median-shift is a mode seeking algorithm that relies on computing the median of local neighborhoods, instead of the mean. We further combine median-shift with Locality Sensitive Hashing (LSH) and show that the combined algorithm is suitable for clustering large scale, high dimensional data sets. In particular, we propose a new mode detection step that greatly accelerates performance. In the past, LSH was used in conjunction with mean shift only to accelerate nearest neighbor queries. Here we show that we can analyze the density of the LSH bins to quickly detect potential mode candidates and use only them to initialize the median-shift procedure. We use the median, instead of the mean (or its discrete counterpart - the medoid) because the median is more robust and because the median of a set is a point in the set. A median is well defined for scalars but there is no single agreed upon extension of the median to high dimensional data. We adopt a particular extension, known as the Tukey median, and show that it can be computed efficiently using random projections of the high dimensional data onto $1D$ *lines, just like LSH, leading to a tightly integrated and efficient algorithm.*

1. Introduction

Mean shift is a popular mode seeking algorithm that is used in a large number of computer vision applications. It is a non-parametric method that does not require a-priori knowledge of the number of clusters, nor does it place any limitations on the shape of the clusters.

Data points belong to the same cluster if they converge to the same mode. However, initializing mean shift from every data point is computationally expensive because each mean shift iteration requires numerous nearest neighbor searches. The problem is exacerbated when dealing with large scale data sets and one often resorts to approximate nearest neighbor tools such as Locality Sensitive Hashing (LSH) to accelerate the process.

We show that LSH can provide more than nearest neighbor query acceleration. LSH works by hashing high dimensional points to bins based on locality. Therefore, points in high density regions will be hashed to the same LSH bin with high probability. We take advantage of this fact to quickly find an excellent initial guess as to the location of the modes, even before applying the mode seeking algorithm. Specifically, We take all high density LSH bins and replace them with their median. This creates a set of representative points, much smaller than the original data set. Next, we shift each point toward the median of its neighborhood. This is similar to medoid shift, but instead of using the point in the set closest to the mean (i.e., the medoid), we use the median that we restrict to be a point in the set. We iterate this procedure until convergence.

This mode detection procedure is faster than mode seeking in detecting the modes of the distribution but, unlike a typical mode seeking algorithm, it does not assign each data point to a mode. To do that we can propagate cluster labels from the modes to the rest of the points. In fact, in applications where only mode detection is needed and not clustering, this step is not necessary.

Switching from mode seeking to mode detection followed by propagation offers some additional advantages as well. Mode seeking is a bottom up approach that is prone to get stuck at saddle points. This is why it is often necessary to perform a post-processing stage to eliminate small clusters. On the other hand, mode detection quickly finds the most significant modes of the distribution and then propagates them to the rest of the data in a top-down fashion. As a result, we do not have to perform any post-processing steps to reason about small clusters or saddle points.

Working with LSH accelerates performance but can also introduce outliers (i.e. points that are not in the neighborhood of the query point). There are two reasons for that. The first is due to the random nature of LSH, and the second has to do with the nature of the data. High dimensional data sets are often assumed to lie on a low dimensional manifold embedded in the high dimensional ambient space. Since LSH operates in the ambient space and not on the manifold, it will return nearest neighbor points according to Eu-

clidean distance and not geodesic distance. This problem arises in mean shift as well and can be dealt with by using M-estimators. However, these estimators are slow to compute. We found the median-shift to be a good compromise between the need for speed and the need for robustness. In addition, median-shift is constrained to land each shift on a point in the set. This has two advantages. First, it can never drift off the manifold and second, we do not have to specify a threshold parameter to determine when mean shift trajectories are close enough to be clustered together.

Working with medians begs the question: what is the definition of a high dimensional median? As it turns out, there are a number of possible extensions proposed in the literature and we use the Tukey median definition. The Tukey median can be well approximated with multiple random projections, just like LSH, which leads to a tightly integrated, fast and robust mode detection and seeking algorithm.

The Tukey median is based on the Tukey depth, which is part of a large class of statistical depth order functions that attempts to define order (i.e., depth) on high dimensional data sets. As a by-product of our work, we demonstrate that the Tukey depth can also be used for saliency detection.

In this paper we show how median-shift manages to find a natural classification in synthetic data-sets where mean shift and medoid shift fail, and demonstrate it in a number of applications.

2. Related work

Previous work on clustering is vast and we only cover here work directly related to ours.

Mean shift was introduced by Fukunaga and Hostetler [9], formalized by Cheng [3] and introduced to the computer vision community by Comaniciu and Meer [4]. Mean shift was first used in Euclidean space and later extended to work on analytic manifolds [18].

Recently, Sheikh *et al.* [17] proposed medoid shift, which applies mean shift to graphs. To do this, they define the medoid to be the point in the data set that is closest to the mean. The result is a mode seeking algorithm that is constrained to move only along points of the set (as opposed to mean shift that can shift to points outside the sample set). This algorithm is reduced to matrix multiplication and as such suffers from a high computational cost of $O(n^{2.38})$ where n is the number of sample points. It was later shown by Vedaldi and Soatto [21] that in case the distance metric is Euclidean the complexity drops to $O(n^2)$.

Georgescu *et al.* [10] combined mean shift and LSH for texture classification. However, their approach only replaced the nearest neighbor search needed by mean shift, with an LSH based search which improved performance considerably. We differ in a number of crucial aspects. First, we propose median-shift which is constrained to points in the data set only (as opposed to mean-shift). This is more robust to outliers than mean shift, as we demonstrate in the experimental section of the paper. In addition, median-shift shares the LSH data-structure which leads to better integration and better performance. Finally, and most importantly, we use LSH to quickly find an initial guess of the location of the modes, which can then be refined using median-shift.

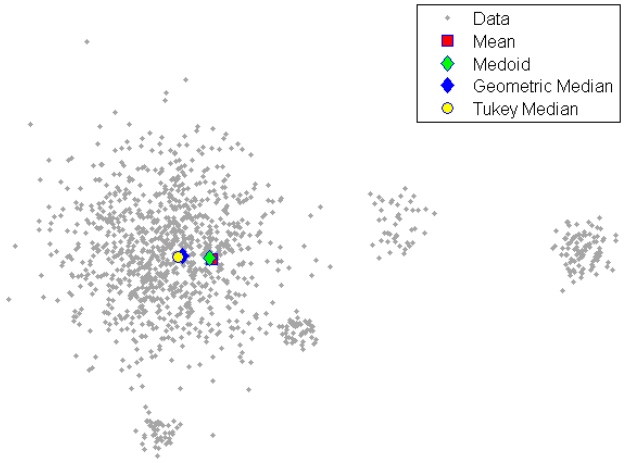

Figure 1. The Tukey median is more robust to noise than either the mean or the medoid. It achieves similar results to the geometric median, however it is faster to compute.

There has also been much work on accelerating mean shift, mainly for image segmentation, using Newton iterations [1], downsampling [5] or by explicitly representing the underlying density function [14].

Switching from mean shift to median-shift forces us to adopt an extension of the median to high dimensional data. A straight forward extension is the geometric median. The geometric median of a discrete set of sample points in a Euclidean space is the point minimizing the sum of distances to the sample points. In other words, the median minimizes the L_1 norm, as opposed to the mean that minimizes the L_2 norm (i.e., the sum of *squared* distances). There is no closed-form solution for finding the geometric median and it does not have to be a point in the set. Within the context of image filtering (i.e. applying median filter to, say, RGB images), the median filter is extended to the vector median filter, which is the point in the data set that minimizes the Euclidean distance to all other points. The geometric median reduces to the standard definition of the median for scalars, but because of its computational cost we explore other extensions (See Figure 1).

The median belongs to a class of statistical depth order functions that attempt to define order on data sets. That is, the goal is to define which points are in the center of the distribution and which points are further away. There are several possible extensions and all reduce to the standard

definition of the median for scalars.

The Convex hull peeling [8] method computes the convex hull of the data set and the assign points of the convex hull a depth order of 1. These points are peeled from the data set and a new convex hull is computed on the remaining points and the points that belong to it are assigned depth order 2, and so on and so forth. The peel median are the points with the highest depth order. This method is attractive for low dimensional data but computing the convex hull of high dimensional data becomes prohibitively expensive, and it is not robust to outliers.

Another approach is the simplicial median [12] that is also defined in terms of the convex hull. A simplicial depth of a point is the probability that this point is in the convex hull of $d+1$ points (where d the dimensionality of the data), chosen independently according to the underlying probability distribution function. The simplicial median is the point with the highest simplicial depth. The sample estimate of the median is the point that is contained in the largest number of simplicials. The reader is referred to [16] for further discussion on depth order functions.

Here we adopt the Tukey median that was proposed by Tukey [20] and rediscovered by computational geometers [15]. The Tukey median of a set of points is a point in the set that maximizes the Tukey depth, where a Tukey depth of a point is the minimum number of sample points on one side of a hyperplane going through the point (see Figure 2). It was recently shown that good approximation of the Tukey median can be achieved using random $1D$ projections of the data [6]. The use of of random projections fits nicely with Locality Sensitive Hashing (LSH). LSH is a randomized algorithm that can be applied by hashing the projection of the the data points on $1D$ lines. Points that are hashed to the same bin are taken to be close in the input, high dimensional, space [7].

3. Preliminaries

We give the preliminary technical details needed to make the paper self contained.

3.1. Mean Shift

Given a set of points $X = \{x_1, \ldots, x_n\}$ in R^d Euclidean space, the kernel density estimate of the point x is taken to be:

$$\hat{f}_k(x) = C \sum_{i=1}^{n} k(\frac{||x - x_i||^2}{h^2}) \tag{1}$$

with bandwidth h, profile function k (i.e., a Gaussian or uniform kernel) and a constant C that ensures that $\hat{f}$ integrates to 1. Letting $g(x) = -k'(x)$ and taking the gradient of (1) we obtain:

$$m_h(x) = \frac{\sum_{i=1}^{n} g(\frac{||x-x_i||^2}{h^2})x_i}{\sum_{i=1}^{n} g(\frac{||x-x_i||^2}{h^2})} - x \tag{2}$$

where, $m_h(x)$ is the mean shift vector which is proportional to the normalized density gradient estimate. Repeatedly applying the mean shift operator is a hill-climbing technique to the nearest stationary point of the density, i.e., a point in which the density gradient vanishes.

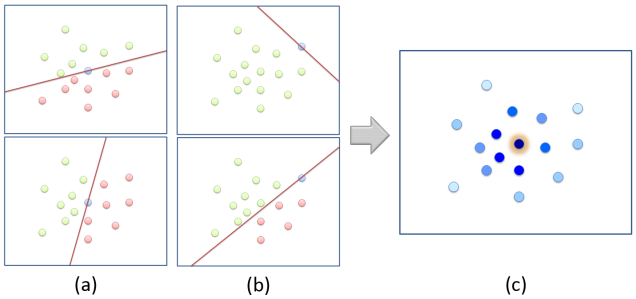

Figure 2. The Tukey median is calculated by passing all possible hyperplanes through every point in the data set. For each hyperplane we take the minimum number of points on either side of the hyperplane. The Tukey depth of a point is the minimum achieved over all hyperplanes. The Tukey Median is the point with the highest Tukey depth.

3.2. Tukey Median

We work our way to the Tukey median of high dimensional points by first revisiting the necessary definitions in 1D and then extending them to high dimensional data.

3.2.1 Depth order of 1D points

Suppose we are given a set $S = \{s_1, \ldots, s_n\}$ of *reals* $s_i \in R^1$, define the *rank* of s_i by

$$RANK(s_i) \equiv |\{s_j : s_j \leq s_i\}| \tag{3}$$

and its *depth* by

$$DEPTH(s_i) \equiv \min(RANK(s_i), n + 1 - RANK(s_i)). \tag{4}$$

The *median* of S is an element with maximal depth:

$$MEDIAN(S) \equiv \max_{s_i} DEPTH(s_i) \tag{5}$$

3.2.2 Depth order of high dimensional points

Given a set $S = \{s_1, \ldots, s_n\}$ of *points* $s_i \in R^d$, $d > 1$, define the Tukey depth of a point to be:

$$DEPTH(s_i) = min[v^T s_i : v \in R^d, ||v|| = 1] \tag{6}$$

That is, the Tukey depth of a point is the minimum of its depth, along any projection vector v. The Tukey median is the point with the highest Tukey depth. There is an optimal randomized algorithm for finding the Tukey median [2] using a variant of linear programming. Unfortunately, it is exponential in d, the dimensionality of the data. A good approximation was reported using a finite number of random projections $V = \{v_1, \ldots, v_k\}$ [6] (see Figure 2). Formally:

$$MEDIAN(S) = \max_{s_i \in S}\{\sum_{v_k \in V} DEPTH(s_i^T v_k)\} \quad (7)$$

3.3. Locality Sensitive Hashing (LSH)

LSH on p-stable distributions works by hashing points in R^d into hash families, such that given two points p and q:

$$\begin{array}{rcl} Pr(H(p) <> H(q) | \, \|p-q\| < r) & \leq & c_1 \\ Pr(H(p) = H(q) | \, \|p-q\| \geq r) & \geq & c_2 \end{array} \quad (8)$$

where r is the parameter, signifying the radius of the R-NN queries, the LSH in effect answers. We define each of the L hash families by randomly selecting k vectors, drawn from R^d normally (for L_2 metric). We define a function $h_{a,b}$ for each projection as:

$$h_{a,b}(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor \quad (9)$$

Where b is drawn uniformly from the range $[0..w]$. Together, the k vectors define a hash function $R^d \rightarrow n^k$. w is selected to minimize the number of false negatives in the returned queries. For a complete description of the algorithm, see [7]. Both LSH and the Tukey median use the same machinery of random projections which leads to a tightly integrated and efficient algorithm.

4. Mode Detection

In this section we describe *median-shift*, a robust and scalable mode detection algorithm. The algorithm consists of shifting each point toward its local neighborhood *median*, using a multi-dimensional depth function, which gives a robust data density estimation. Recall that the mean and median coincide for symmetric distributions (i.e., Gaussian or uniform distributions) and hence as the number of samples grow, the median-shift converges to the density mode. In the following subsections we present the theoretical background, as well as the implementation details.

4.1. Median-Shift Mode Seeking

The median-shift algorithm is similar to other mode-seeking algorithms such as mean shift and medoid shift. For each point we wish to ascend in the direction of the positive gradient of the underlying probability density function. This is equivalent to moving toward the local representative of the neighborhood, in our case, the Tukey median. Therefore we define the median-shift for point c in set P as:

$$c' = MEDIAN(\{p \in P | \, \|p - c\| \leq r\}) \quad (10)$$

where r is the bandwidth parameter of the algorithm. Since c' is necessarily a point in the dataset, there is no need for multiple iterations in this step (similar to medoid shift, but unlike mean shift). After one iteration all points are linked and we can only go through the list of discovered medians to find a mode.

The results of this step are a set of modes representing clusters. We proceed by iteratively working on the reduced set of modes, replacing the median calculation by weighted median calculation, where the weights are the number of points mapped to the given mode. We stop the iterations when the set of modes does not change from one iteration to the next.

4.2. Significant Mode Detection

Running median-shift on all data points is possible but is usually time consuming. The basic operation in this procedure is calculating the depth of points in the neighborhood of each point and sorting them. Although LSH gives a good approximation of the neighborhood, it still takes time to sort all neighborhood points especially in high density areas of the data, i.e. in bins that contain many points. In fact, we found that most of the time in a mode seeking algorithm is spent on sorting the same LSH bins in high density areas again and again. This inspired us to examine those bins.

We found (Figure 3) that in all projections the distribution of bins is similar. A small number of bins (in each projection) cover most of the data. Since significant modes in the data will most likely fall within these high density areas and all points in such areas have a high probability of being hashed to the same mode, we replace each bin with its median as a representative point (Figure 4 (b)).

Next, we run the median-shift procedure on the set of representative points iteratively until convergence. In each iteration we replace the previous set with the set of detected modes, weighted by the number of points mapped to them (Figure 4 (c)). These weights are taken into account during the calculation of the depth of each point in the next iteration by modifying the definitions as follows:

$$RANK(s_i) \equiv \sum_{s_j \leq s_i} \text{Weight}(s_j) \quad (11)$$

and defining $W = \sum_{s_j} \text{Weight}(s_j)$ as the total weights in the neighborhood of s_i then:

$$DEPTH(s_i) \equiv \min(RANK(s_i), W - RANK(s_i)). \quad (12)$$

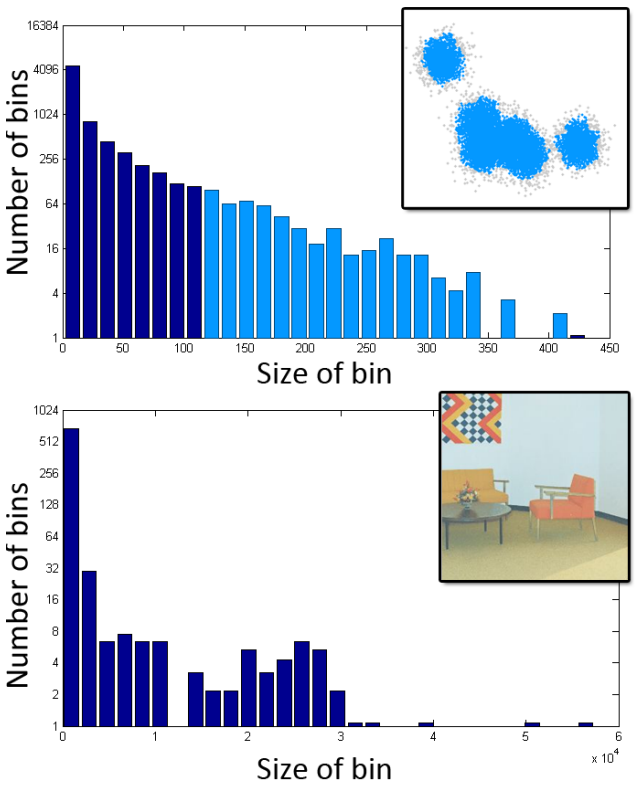

Figure 3. Histograms of point density of LSH bins (y axis in log scale). The top example shows bin distribution on synthetic data. The 500 largest bins (out of 7000) cover 91% of the data (highlighted in light blue). The lower example shows bin distribution on a 3x3 patch feature space of an image. Both histograms demonstrate that a small number of bins contain a significant portion of the data. These bins serve as an excellent initial guess for mode detection as they cover densely populated regions.

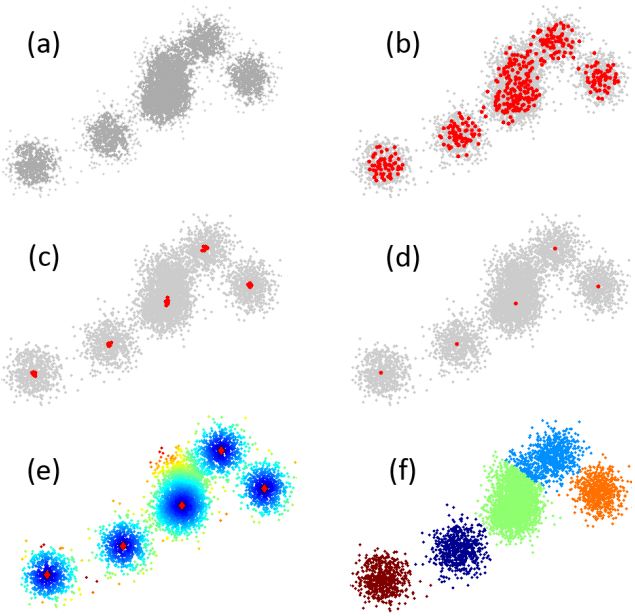

Figure 4. Significant mode detection by LSH. (a) Original data set (b) Tukey medians of the largest bins in the LSH (c) Single median-shift step for each initial representative converges to a few modes(d) We iterate over the modes until convergence using a weighted median-shift step (e) Using a geodesic propagation step we assign a mode to each point (f) Final clustering results.

	LSH	kd trees	Dim.	N
Construct	16.7 sec	52.3 sec	147	246K
Query	3.6e-4 sec	0.019 sec	147	246K

Table 1. Comparison of LSH and kd-trees. We compare the problem of clustering $246,000$ points living in a 147 dimension space. LSH is faster in construction and query time, over kd-trees.

Finally, in case of data clustering, and not only mode detection, we map each data point to its closest mode. This is done using multiple seed breadth-first-search based on the geodesic distance in the data set (Figure 4 (e)).

5. Experiments

We have implemented median-shift in C++ with a matlab interface. We then performed a couple of experiments to evaluate it.

5.1. Synthetic experiments

First, we compared the performance of LSH to the kd-tree based ANN library [13]. Our dataset consisted of $246,000$ image patches of size $7 \times 7 \times 3$ (equivalent to 147 dimensions). We measure the time it takes to construct a database and the time it takes to query the database. As can be seen in Table 1, LSH is about four times faster in constructing the dataset and querying it.

In another experiment, we compare four types of mode seeking algorithms: median shift with mode seeking or mode detection, medoid shift and mean shift with mode seeking. As is often suggested in the literature, we terminate mode seeking iterations if we encounter a point that is already assigned to a cluster. Table 2 reports the results. It shows that mode detection runs up to 10 times faster than a mode seeking approach that is applied to every point in the data set (even with early trajectory termination).

Next, we compared mean shift, medoid shift and median shift on a synthetic data set shwon in Figure 5. This is a challenging example because it shows the possible confusion of Euclidean and geodesic distance. All algorithms fail to cluster the data into only two clusters, because the data is not organized in two, nicely clustered Guassians. As one increases the bandwidth, both mean shift and medoid shift break down and cluster points from the two sine waves together. Median shift, on the other hand, proves to be more robust. It never clusters points from the two sine wave to-

num	dim	mseek	mdetect	med.	mean
5000	2	2.736	0.84	15.2	1.341
10000	2	14.6	2.88	na	2.07
30000	2	147.9	13.74	na	8.1
5000	10	10.13	1.38	na	3
5000	20	24	3.5	na	2.7

Table 2. We ran several experiments to demonstrate the speedup of mode detection over mode seeking. The same data set was clustered using four algorithms: mode seeking via median-shift, mode detection via median-shift, medoid shift and mean shift. Mode detection via median-shift shows great speedup compared to mode seeking, and is comparable in performance to simple mean shift. Medoid shift could not be run on more than 5000 points due to the necessity of calculating a distances matrix n^2.

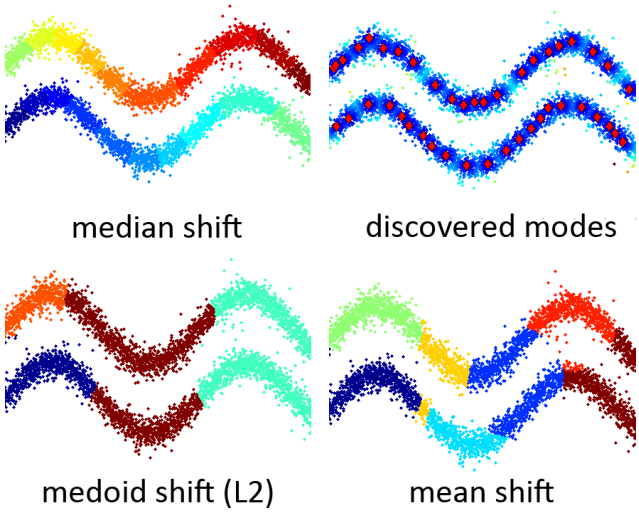

Figure 5. A comparison of median-shift to Medoid shift and Mean shift on a synthetic data set. From top left, in a clockwise direction. The result of median-shift clustering, the initial set of mode candidates detected using LSH bin probing, result of mean shift clustering, and result of medoid shift clustering. Median-shift is more robust than either mean shift or medoid shift. It never clusters points from the two sine waves together. (All mode seeking algorithms struggle in uniform regions, as demonstrated here).

gether.

We also demonstrate that median shift works properly on manifolds. Figure 6 applies our algorithm to the data set proposed by Sheikh *et al.* [17]. We achieve the same clustering results at a fraction of the time.

5.2. Saliency detection using the Tukey depth

Before we move to experiments of the median shift on real data, we highlight the capabilities of the Tukey depth as a depth order function for the problem of saliency detection. Given an image, we break it into overlapping 7×7 pixel patches and want to find the most salient ones. We

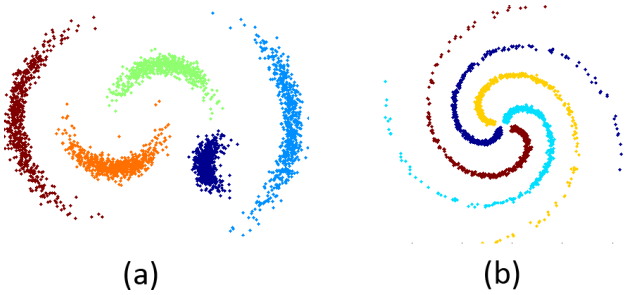

Figure 6. Medoid shift on manifolds. We demonstrate median-shift on the data sets proposed in [17], results are comparable and are achieved in a fraction of the time.

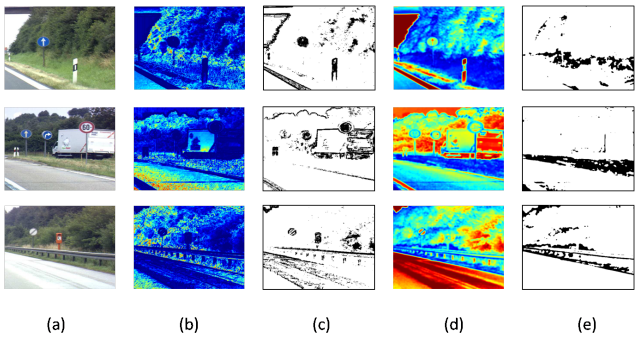

Figure 7. Saliency detection using Tukey depth. We break each image into overlapping 7×7 pixel patches and compute their Tukey depth. The bottom 1% Tukey depth patches are taken to be salient patches. For comparison, we repeat this experiment using the mean, instead of the Tukey depth. (a) The original image. (b) Tukey depth (darker blue means lower value). (c) Thresholded Tukey depth of bottom 1% (shown in black). (d) Distance from mean patch. (e) Thresholded mean patch distance (we threshold the mean image to have the same number of pixels as (c)). Clearly, Tukey depth captures salient regions much better.

interpret this to be patches with a very *low* Tukey depth. Figure 7 show the results on several images from the traffic sign database [11]. For comparison we repeat the process with the mean, instead of the Tukey depth. That is, we first compute the mean patch and then measure the distance of all patches to the mean patch, where distance is taken to be the sum of absolute pixel intensity values differences. Clearly, the Tukey median does a much better job.

5.3. Image segmentation

We applied median-shift to image segmentation and compare two possible representations. In the first case, each pixel is represented by a simple $xyrgb$ representation where the XY coordinates are scaled according to an approximate desired spatial bandwidth. In the second representation each pixel is represented by a combination of a 3×3 patch surrounding it and its coordinates. In both cases, we do not rely on image coordinates to accelerate performance. Instead we rely on LSH for fast nearest neighbor queries

Figure 8. Image segmentation. (a) Original image (b) Segmentation overlay (c) Simplified image (d) Manual results using EDISON mean shift. Images i,ii,v were segmented using median-shift on 3x3 patches. Images iii,iv,vi were segmented using median-shift on RGB.

and for initializing the location of the modes. We show several typical results in Figure 8.

Segmenting a 250×300 image in $xyrgb$ takes 13.1 seconds, while using local patches (29 dimensions) takes 16.2 seconds. Skipping the mode detection step and initializing the mode seeking procedure from every pixel can take more than a minute to complete, even when using early trajectory termination (i.e., we stop iterations if we encounter a pixel that is already assigned to a cluster). For comparison we measured timing performance of the mean-shift based EDISON system [5] that does rely on the structure of the image to limit the range of nearest neighbor queries. We hand tuned EDISON to give results as similar as possible to ours and found that it takes 12.7 seconds with no speedup and 4.5 with speedup enabled.

5.4. Chromatic noise filtering

Chromatic image noise affects the direction (chromaticity) of the color vector but not its intensity. Unit vectors lie on a manifold and hence standard mean shift does not apply. One can use nonlinear mean shift over Riemannian manifolds [19], or use median shift as we show here.

Given an input image, we add noise by changing the direction of the color vectors, but not their length. We then compare two methods for chromatic noise reduction. The first method relies on the Tukey median filter. Specifically, we go over every 7×7 neighborhood and select the Tukey median over the normalized RGB vectors. After filtering the whole image we restore the original vector lengths to get the final image. We varied the neighborhood size from 3×3 up to 9×9 with similar results. The second method relies on mode detection using median shift. We first normalize RGB pixels to unit length and then map all pixels to $rgbxy$ feature space, where the xy component is scaled appropriately to approximate a spatial bandwidth. We then construct the LSH but use only the normalized rgb component, and not the xy. This way, the Tukey median is computed on local neighborhoods of the normalized rgb component, discarding the xy component.

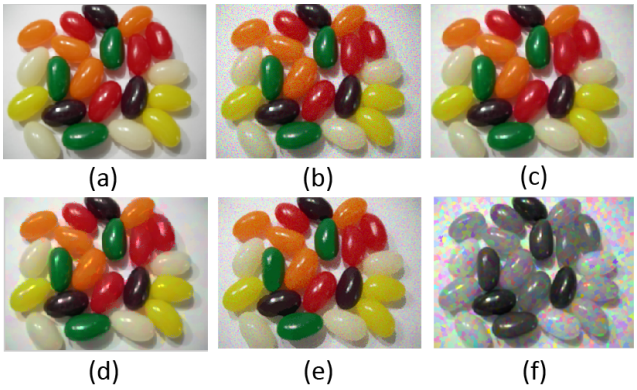

Figure 9. Chromatic noise filtering. (a) Original image. (b) Noisy image. (c) Tukey median filter. (d) Median shift using Mode detection. (e) Result using EDISON mean shift. (f) Mean shift using Mode detection. See text for more details.

Mean-shift segmentation using EDISON is unable to remove the noise, either leaving the image noisy, or blurring it. Applying mode detection via mean-shift to the image, on the normalized color vectors also fails. In this case, the mean of a local neighborhood is not constrained to remain on the manifold, and thus fails to achieve a good result (See Figures 9 and 10).

6. Conclusions

Median shift is a mode seeking algorithm that is based on the median, instead of the mean. It is coupled with LSH for fast nearest neighbor queries which makes it suitable for handling large scale data sets. A novel contribution of our work is the mode detection step that greatly accelerates performance. Instead of initializing median shift from every point, we analyze the point density of the LSH bins and use the ones that are most dense as an initial guess for mode seeking. We then propagate the cluster labels from the modes to the rest of the data points. This top-down approach, as opposed to the bottom-up mode seeking ap-

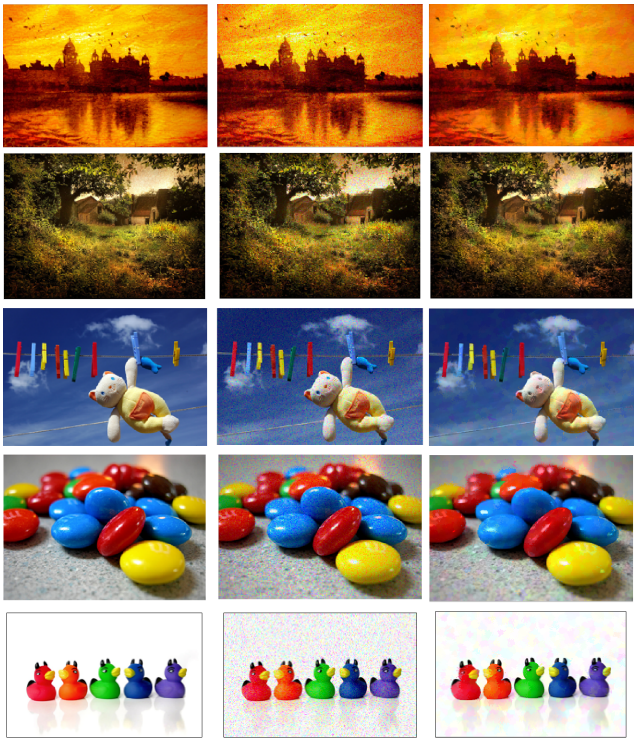

Figure 10. Additional results in chromatic noise filtering. Original image on left, noisy image in the middle, median shift noise reduction on the right.

proach, does not require post-processing to reason about saddle points or small cluster. We compute high dimensional medians using the Tukey median and show that it can be efficiently approximated using random $1D$ projections, just like LSH, leading to a tightly integrated and efficient algorithm. We demonstrate the algorithm on both synthetic and real data, including image segmentation and chromatic noise filtering.

References

[1] M. A. Carreira-Perpinan. Acceleration strategies for gaussian mean-shift image segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2006. 2

[2] T. M. Chan. An optimal randomized algorithm for maximum tukey depth. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 430–436, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics. 4

[3] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995. 2

[4] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1197, Washington, DC, USA, 1999. IEEE Computer Society. 2

[5] D. Comaniciu and P.Meer. A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis Machine Intelligence (PAMI)*, 2002. 2, 7

[6] J. A. Cuesta-Albertos and A. Nieto-Reyes. The random tukey depth. *Comput. Stat. Data Anal.*, 52(11):4979–4988, 2008. 3, 4

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM. 3, 4

[8] W. Eddy. Convex hull peeling. *COMPSTAT*, pages 42–47, 1982. 3

[9] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975. 2

[10] B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 456–463 vol.1, 2003. 2

[11] http://ilab.usc.edu/toolkit/downloads.shtml. 6

[12] R. Liu. On a notion of data deph based on random simplices. *The Annals of Statistics*, 18:405–414, 1990. 3

[13] D. M. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching. 1997. 5

[14] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. 2

[15] M. S. R. Cole and C. Yap. On k-hulls and related topics. *SIAM J. Comput.*, 16:61–77, 1987. 3

[16] R. J. S. R. Liu and D. L. Souvaine. *Data Depth: Robust Multivariate Analysis, Computational Geometry, and Applications*. AMS Bookstore, 2007. 3

[17] Y. Sheikh, E. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *ICCV 2007: Proceedings of the 11th International Conference on Computer Vision*, pages 1–8, Washington, DC, USA, Oct 2007. IEEE Computer Society. 2, 6

[18] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *CVPR 2006: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1168–1175, Washington, DC, USA, 2006. IEEE Computer Society. 2

[19] R. Subbarao and P. Meer. Nonlinear mean shift over riemannian manifolds. In *To appear in International Journal on Computer Vision*, 2009. 7

[20] J. Tukey. Mathematics and the picturing of data. 1971. 3

[21] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008. 2