Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

Efficient Mean-shift Clustering Using Gaussian KD-Tree

Chunxia Xiao Meng Liu

The School of Computer, Wuhan University, Wuhan, 430072, China

Abstract

Mean shift is a popular approach for data clustering, however, the high computational complexity of the mean shift procedure limits its practical applications in high dimensional and large data set clustering. In this paper, we propose an efficient method that allows mean shift clustering performed on large data set containing tens of millions of points at interactive rate. The key in our method is a new scheme for approximating mean shift procedure using a greatly reduced feature space. This reduced feature space is adaptive clustering of the original data set, and is generated by applying adaptive KD-tree in a high-dimensional affinity space. The proposed method significantly reduces the computational cost while obtaining almost the same clustering results as the standard mean shift procedure. We present several kinds of data clustering applications to illustrate the efficiency of the proposed method, including image and video segmentation, static geometry model and time-varying sequences segmentation.

Categories and Subject Descriptors (according to ACM CCS): I.4 [Computing methodologies]: Image Processing and Computer Vision—Applications

1. Introduction

Mean shift is a well established method for data set clustering, it has been widely used in image and video segmentation [CM02] [WTXC], object tracking [CRM03], image denoising [BC04], image and video stylization [DS02] [WXSC04], and video editing [WBC*05], it also has been extended to geometry segmentation [YLL*05] and 3D reconstruction [WQ04]. Mean shift works by defining a Gaussian kernel density estimate for underlying data, and clusters together the points that converge to the same mode under a fixed-point iterative scheme. Although mean-shift works well for data clustering and obtain pleasing clustering results, however, the high computational complexity is the one of main difficulties to apply mean shift to cluster large data set, especially for those situations where the interactive and even real time clustering processing are preferred.

The complexity for the standard mean shift procedure is $O(\tau d n^2)$, where n is the number of the data points, the τ is the number of the iterations for mean shift clustering procedure, and d is the dimension of the point. The most expensive computing is to find the closest neighborhood for each point in the data space, which is a multidimensional range searching method. Even using one of the most popular nearest neighbor search method, the ANN method [AMN*98], given a query point q and $\varepsilon > 0$, a $(1+\varepsilon)$ approximate nearest neighbor of q must be computed in $O(c_{d,\varepsilon} \log n)$ time, where $c_{d,\varepsilon}$ is a factor depending on dimension d and ε. Therefore, when processing large data sets, the high time complexity leads to serious difficulty. Although many acceleration techniques have been proposed [EDD03, GSM03, YDGD03, WBC*05, CP06, PD07, FK09], further improvements are still desirable for both performance and clustering quality.

In this paper, inspired by the fast high-dimensional filtering method using Gaussian KD-trees [AGDL09], we propose an efficient paradigm for mean-shift procedure computing. Our method is based on following key observation, since the mean shift procedure clusters those points that are feature similar, while there are many clusters of points which are high similar in feature, it is wasteful to perform mean shift procedure for each point to converge to the mode. Thus we first cluster the original point set into clusters based on feature similarity using KD-tree [AGDL09], and obtain the Gaussian weighted samples of the original data set, which is the reduced feature space for approximating original point set. Then instead of computing mean shift directly on original individual points, we compute on the samples (which is of a much smaller number) to obtain the modes of the sample space. Finally we find the closest mode for each point based on Gaussian weighted feature similarity, and construct the final clustering results.

As mean shift is performed on a greatly reduced space (typically thousands of times smaller than original data set), and all stages of our algorithm are data-parallel across queries and can be implemented the algorithm in CUDA [Buc07], we can cluster the large data set in real time or at interactive rate (for a video with 1.44×10^7 pixels in Figure 5). Furthermore, as the sample space is an approximate feature space of the original data set generated using the proposed Gaussian weighted similarity sampling, our method receives accurate results comparable with the standard mean shift that performed on the original data set. In addition, our method uses only an extremely small fraction of resources, for both time and memory consuming.

This paper is organized as follows. In section 2, we give the related work, section 3 is the main part of our paper, we describe the proposed fast mean shift clustering method. In section 4, we give the applications of our method and comparisons with the related mean shift acceleration methods, and we conclude in section 5.

2. Related work

Mean shift was first presented by Fukunaga and Hostetler [FH75], and it was further investigated by Cheng et al. [Che95] and Comaniciu et al. [CM02]. Mean shift is now a popular approach for data set clustering, and has been widely used in image and video segmentation [CM02] [WTXC], object tracking [CRM03], image denoising [BC04] and image and video stylization [DS02] [WXSC04], it also has been extended to geometry segmentation [YLL*05] and 3D reconstruction [WQ04], and many image and video editing methods are based on the mean shift clustering preprocessing [WBC*05].

One of the main difficulties in applying Mean Shift based clustering to large data sets is its computational complexity. For each Gaussian weighted average iteration, the complexity of brute force computation is quadratic in the number of data points. There are several existing techniques which have been developed to increase the speed of Mean Shift. Comaniciu and Meer [CM02] used axis-aligned box windows, however, this produces many limit points and adjacent points are merged as a post-process. Dementhon [DeM02] used multiscale structure to accelerate video segmentation. Yang at al. [YDDD03] applied the Fast Gauss Transform to speed up the sums of Gaussians in higher dimensions that were used in the Mean Shift iteration. This method is effective for Gaussian weighted average with large filtering radius, however, performing weighted average in a relative small radius does not benefit much from this method. Georgescu et al. [GSM03] accelerated mean shift by performing fast nearest neighbor search with spatially coherent hash tables. Carreira-Perpinán [CP06] studied four acceleration strategies and found that spatial discretization method (using uniform down sampling schemes) performed best.

Paris and Durand [PD07] applied the sparse representation of the density function to accelerate mean shift, similar to the bilateral filtering [PD06], they first binned the feature points in a coarse regular grid, and then blurred the bin values using a separable Gaussian. The computational complexity and memory scale exponentially with the dimension d. Wang et al. [WLGR07] used a dual tree to speed up Mean Shift by computing two separate trees, one for the query points, and one for the reference points. Compared to the methods of [YDDD03] [PD07], this method maintains a relative error bound of the Mean Shift iteration at each stage, leading to a more accurate algorithm, however, the performance of this method is much lower than [YDDD03] [PD07]. More recently, Freedman and Kisilev [FK09] proposed a sampling technique for Kernel Density Estimate (KDE), they constructed a compactly represented KDE with much smaller description complexity, this method greatly accelerates the mean shift procedure, however, the accuracy of the mean shift clustering depends on the number of the random samples.

Many methods have applied Gaussian KD-Trees for accelerating image and video processing. Adams et al. [AGDL09] applied Gaussian KD-Trees for accelerating high-dimensional filtering includes the bilateral image filter [TM98], bilateral geometry filtering [JDD03, FDCO03] and image denoising with nonlocal means [BCM05]. We borrow some ideas from [AGDL09] for adaptive clustering in this paper. Xu et al. [XLJ*09] used K-D Tree to build adaptive clustering for accelerating affinity-based image and video edit propagation. As an alterative, Xiao et al. [XNT10] used quadtree based hierarchical data structure to accelerate edit propagation. KD-Trees have been widely used in accelerating graphics rendering [HSA91], the real-rime KD-tree construction on graphics hardware also have been proposed [HSHH07] [ZHWG08]. Our method applies Gaussian KD-Trees [AGDL09] to build a hierarchy and clustering for the large data set to accelerate the mean shift computing. Compared with Paris and Durand [PD07], whose complexity is exponential in the dimension d of the point, our tree-based mean shift provides with excellent performance, as its runtime and memory consuming both scale linearly with dimension of the points. With the samples generated using similarity-based KD-tree clustering, our method obtains more accurate results than [FK09] when using similar number of samples.

3. Fast mean shift clustering

We first give brief review of mean shift, then we describe the proposed fast mean shift clustering method, including data set clustering preprocess using KD-tree, sample feature space computing, mean shift modes computing in reduced feature space, modes interpolation. We also present the complexity analysis and GPU implementation of the proposed algorithm.

3.1. Review of mean shift

Given point data set $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$ is d dimensional feature vector, each is associated with a bandwidth value $h_i > 0$. An adaptive nonparametric estimator of this data set is defined as

$$f_K(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}k\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right) \tag{1}$$

where $K(x) = c_{k,d}k\left(\|x\|^2\right) > 0$ is kernel function satisfying $K(x) \geq 0$ and $\int_{\mathbb{R}^d} K(x)dx = 1$, By taking the gradient of (1) the following expression can be obtained.

$$\nabla f_K(x) = \frac{2c}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)m(x) \tag{2}$$

where $g(x) = -k'(x)$, and $m(x)$ is the so-called mean shift vector

$$m(x) = \frac{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}x_i g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)} - x \tag{3}$$

The expression (3) shows that at location x the weighted average of the data points selected with kernel K 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 following gradient-ascent process with an adaptive step size until convergence constitutes the core of the mean shift clustering procedure

$$x_{i+1}^* = \frac{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}x_i g\left(\left\|\frac{x_i^*-x_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g\left(\left\|\frac{x_i^*-x_i}{h_i}\right\|^2\right)}, \quad j = 1,2,\ldots \tag{4}$$

The starting point of the procedure x_i^* can be chosen as points x_i, and the convergence points of the iterative procedure are the modes of the density. The all points that converge to the same mode are collected and considered as a cluster. More details are described in [CM02].

3.2. Fast mean shift computing

The weighted average of expression (4) is the most time consuming computing of mean-shift when the number n of the data set is large (for example, 10^9 pixel in video streaming). Given an arbitrary set of point $\{x_i\}_{i=1}^n$ with feature vector of d dimension, a naive computation of mean shift vector expression (4) would take $O(dn^2)$ time, as every point interacts with every other point. A simple way to accelerate the mean shift procedure is using the weighted average of the closest points of x_i, and the bandwidth value h_i can be set depending on the neighborhood size. However, using this scheme, we have to perform the nearest neighborhood search, which is also a time consuming operation for large data set, especially for data set with high dimension vector.

To accelerate the weighted average operation of expression (4), instead of computing expression (4) for individual points in the data set, we approximate original data set by piece-wise linear segments in the feature space based on similarity measure, each represented by a cluster of nearby pixels, and the size of each cluster is adapted to the size of the similar feature space. The generated clusters can be considered as the samples of the data set, which is of a much smaller number than the number of point. Then instead of solving the mean shift procedure (4) directly on individual points as done in previous methods, we solve it on the samples based on Gaussian weighted average on a neighborhood, finally we interpolate the clustering results to the original data set.

We cluster the point data based on similarity measure between the points, which is defined in the feature space of input data set. We define the similarity between the points using both spatial locality p and value v of point, which constitutes the feature space of the input data set. For example, in image case, point x is a pixel with its position $p = (x,y)$ and its color value $v = (r,g,b)$ (Lab color space). Thus, each point x is a five-dimensional feature vector whose axes are $x = (p,v) = (x,y,r,g,b)$. As stated in [AP08] [XLJ*09], the similarity measure (or affinity) between two points can be defined as $z_{ij} = \exp\left(-\|x_i - x_j\|^2\right)$, the position p and value v also can be weighted by parameters. For video, the feature vector can be expanded to include the frame index t and motion estimation ς of point x, and feature vector is expressed as seven-dimensional vector $x = (p,v,t,\varsigma)$. For image, we compute the mean shift clustering procedure (4) in feature space where each point is associated with both spatial locality p and value v.

3.2.1. KD-Tree adaptive clustering

We apply KD-tree to adaptively cluster the data set in the feature space, and subdivide finely in the regions where the point feature vectors are different, subdivide coarsely in the regions where the feature vectors are similar. In image KD-tree clustering, for example, we subdivide the homogeneous regions coarsely, while subdivide the edges regions finely. Then by representing each cluster with a sample, we can receive an accurate approximate feature space of the original data set with much less samples.

KD-tree clusters the data set based on feature space in a top down way. Starting from a root cell, the top rectangular cell represents all points in the data set, we recursively split a cell to two child cells adaptively along a dimension that is alternated at successive tree levels. Similar to [AGDL09], each inner cell of the tree T represents a d-dimensional rectangular cell which stores six variables: the dimension d along which the tree cuts, the cut value T_{cut} on that dimension, the bounds of the cell in that dimension $T_{\min}$ and $T_{\max}$, and pointers to its children T_{left} and T_{right}. Leaf nodes contain only a d-dimensional point which is considered as a sample. This sample represents the points that contained in the leaf cell, the kd-tree stores m samples $\{y_j\}_{j=1}^{m}$ of original data set in d-dimensions, one sample point per leaf. The samples $\{y_j\}_{j=1}^{m}$ construct the reduced feature space of the original data set.

As illustrated in Figure 1, we adaptively cluster the image into clusters. The image is adaptively clustered, where at the edge regions, more samples are placed, while at the homogeneous regions, coarse samples are placed. The sample factor can be changed by the stopping criteria. We present two thresholds for stopping criteria, one is the size of the cell, other one is the variance σ of the similarity measure z_{ij} between the points in the cell. By using these two thresholds we can generate different sampling factor for image as well as respecting for the feature distribution of the data set. Figure 1 shows some cluster results with different sampling factor.

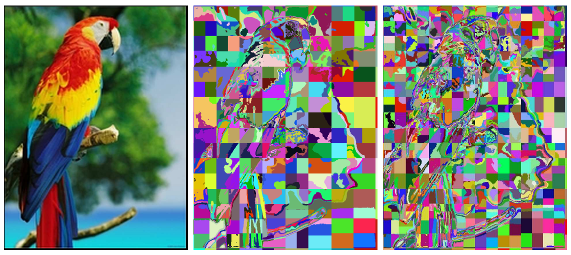

Figure 1: *Image sampling using adaptive KD-tree. (a) Original image* (636×844)*, (b) image clustering with 1,050 samples, (c) image clustering with 11,305 samples.*

3.2.2. Sample feature space computing

We obtain the samples $\{y_j\}_{j=1}^{m}$ in d-dimensions, which are adaptive samples of the feature space of original data set. The samples $\{y_j\}_{j=1}^{m}$ can be considered a approximate feature space of original space. To make a more accurate approximate feature space, we scatter the point x_i to the samples $\{y_j\}_{j=1}^{m}$ based on the affinity similarity z_{ij} between the point x_i and sample y_j, and obtain an affinity similarity based sample space.

Similar to the splatting stages in [AGDL09], we scatter each point x_i to its nearest neighborhood $N(x_i)$ in samples $\{y_j\}_{j=1}^{m}$ (We apply the KD-tree to search the high dimension nearest neighborhood $N(x_i)$ for point x_i), and compute an affinity based sample y_j^* for each sample y_i. The affinity based sample y_j^* can be considered as the weighted similarity average for those feature vector $\{x_i\}$ that is most similar to the sample y_i. We compute and sum the affinity similarity z_{ij} between x_i and each $y_j \in N(x_i)$ to obtain the affinity based sample y_j^* of the sample y_i: $y_j^* = y_j^* + z_{ij}x_i$, then y_j^* is normalized by the sum of the similarity z_{ij}. The generated y_j^* is a feature vector of d-dimensions. The affinity based sample $\{y_j^*\}_{j=1}^{m}$ is a more accurate samples of the original point set, and will be used in the mean shift clustering procedure and modes interpolation. Then for each sample, we store two vectors, one is feature vector y_i, the other is the affinity based sample y_j^*.

3.2.3. Mean-shift modes computing

After obtaining the affinity based samples $\{y_j^*\}_{j=1}^{m}$ for original data set feature space, instead of computing the mean shift clustering procedure in the original space, we compute the mean shift procedure on the reduced feature space $\{y_j^*\}_{j=1}^{m}$. Note for each iteration, we find for each sample y_j^* the nearest neighborhood $N(y_j^*)$ in the sample space, and perform following gradient-ascent process with an adaptive step size until convergence:

$$u_{j+1} = \frac{\sum_{y_i^* \in N(y_j^*)} y_i^* g\left(\left\|\frac{u_j - y_i^*}{h}\right\|^2\right)}{\sum_{y_i^* \in N(y_j^*)} g\left(\left\|\frac{u_j - y_i^*}{h}\right\|^2\right)}, \quad j = 1, 2, \ldots \tag{5}$$

Iterating an infinite number of times the expression (5) is guaranteed to bring u_j to the mode in reduced feature space. Practically, in the implementation, we find that the mean-shift clustering procedure (5) tends to converge in a very small number of steps, typically around 6.

As the number of the samples $\{y_j^*\}_{j=1}^{m}$ is much smaller than the number of the points $\{x_i\}_{i=1}^{n}$, that is $m \ll n$, the computational complexity of mean shift vector computing has been reduced significantly. Furthermore, the mean shift is performed in the affinity similarity based reduced feature space, which leads to more accurate modes. We apply the KD-tree to perform the high dimension nearest neighbor research for each iteration computing. After performing mean shift iterations for each sample y_j^*, we receive the modes $\{z_k\}_{k=1}^{s}$ of the reduced feature space $\{y_j^*\}_{j=1}^{m}$, which are the approximate modes of original data set. All samples converging to the same mode are clustered together.

3.2.4. Modes interpolation

To interpolate the modes computed in the reduced feature space to the original data set, one naïve approach is to find a nearest mode $z_l \in \{z_k\}_{k=1}^{s}$ for each point x_i. This can be considered as a hard clustering. As an alternative method, we give a soft clustering method which generates more smooth clustering results. This method works by applying weighted based interpolation.

The mode interpolation works as follows, for each point x_i, we find the nearest samples $N(x_i)$ in $\{y_j^*\}_{j=1}^m$. Each sample $y_j^* \in N(x_i)$ converges to a mode u_j, that is, $y_j \rightarrow u_j$. Based on the affinity similarity z_{ij} between x_i and y_j^*, by normalizing the weights z_{ij}: $\sum_j z_{ij} = 1$, the final mode are computed as: $x_i \rightarrow \sum_{j=1} z_{ij} u_j$. When we compute the weighted mode interpolation over all the samples $\{y_j^*\}_{j=1}^m$, similar to [FK09], we will receive the cartoon-like clustering results.

Note that in the interpolation stage, the size of the $N(x_i)$ is not always the same as that performed in the sample space computing stage. In our experiment, we set neighborhood size between 10 and 20, and receive satisfied results. As the samples are significantly smaller than the number of the original point set, using the GPU accelerated KD-tree nearest neighborhood search, the search performing is fast. In addition, since we determine the final mode for x_i based on the weighted similarity, the results are more accurate. Figure 3 shows the results using the two different mode selection methods, one is the nearest-sample based mode selection, the other is weighted modes interpolation. As illustrated in Figure 3, using the weighted modes interpolation, we receive smoother and more accurate results.

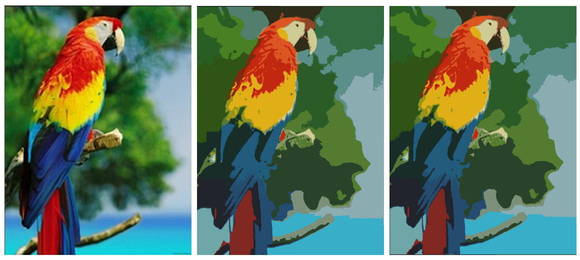
(a) (b) (c)

Figure 2: *Image segmentation results comparison. (a) The original image, (b) the result using nearest sample based mode selection, (c) the result using weighted modes interpolation.*

3.3. Complexity analysis and GPU implementation

Our algorithm accelerates the mean shift procedures by computing a lower resolution feature space and then interpolating the clustering result to the original data set. By using a KD-tree structure, we construct a reduced feature space for input n d-dimensional data points with m feature vector: $\{x_i\}_{i=1}^n \rightarrow \{y_j^*\}_{j=1}^m$, and $m \ll n$. Assuming the depth of Gaussian tree is $O(\log m)$, the complexity of tree construction is $O(nd\log m)$. Performing nearest neighborhood for each of the n input points to scatter values into the tree takes $O(n(\log m + d))$ time. If performing τ iterations for mean shift clustering procedure, computing the mean shift iteration in the reduced space with m feature vector takes $O(\tau m(\log m + d))$ time. In the last stage, we up-sample clustering results to the original data sets which takes $O(n(\log m + d))$ time. Recall that $m \ll n$, this results in a total complexity $O(dn\log n)$. Compared with standard mean shift procedure with complexity $O(\tau dn^2)$, the proposed method significantly accelerated.

Applying the method presented by [ZHWG08], a KD-tree is efficiently built on the GPU for the input points with high-dimensional feature vector. Three stages of our proposed algorithm, including the points scattering, mean shift clustering procedure in the reduced space, and modes interpolation, all incorporate the high dimensional nearest neighbor search. As the high dimensional nearest neighbor search can be implemented in parallel using GPU [AGDL09], thus, our method is even fast to process large data set with high dimensional feature vector. We implement the proposed algorithms in CUDA, and run it on an NVIDIA GeForce GTX 285 (1GB) graphics card. We observe a typical speedup of 20x over our single-threaded CPU implementation running on an Pentium(R) Dual-Core CPU E5200@2.50GHz with 2GB RAM, which allows our method to be applied in an interactive mean shift clustering framework for moderate-sized data set.

4. Applications and comparisons

We apply the proposed fast mean shift clustering to following applications, image segmentation, video segmentation, geometry model segmentation, and animated object segmentation. We also present the comparison results on both performance and segmentation quality with the most related methods. Our approach is implemented using C++ on a machine equipped with Pentium(R) Dual-Core CPU E5200@2.50GHz with 2GB RAM. The GPU acceleration is based on CUDA [`http:` `http://www.nvidia.com/CUDA`] and run on a NVIDIA GeForce GTX 285 (1GB) graphics card.

4.1. Image segmentation

We apply the proposed fast mean shift method for image segmentation. All pixels that converge to same mode are collected together and are considered to be the same segment. In image case, we define the feature vector of pixel $x_i = (\sigma_p p_i, \sigma_c c_i)$ comprising its position $p = (x, y)$ and its color value $c = (r, g, b)$ (Lab space) which can be weighted by parameters σ_p and σ_c. Thus, each pixel x_i is a five-dimensional vector.

Figure 3 presents the segmentation results generated applying our fast mean shift method based on different sampling factor. As illustrated in Figure 3, there are 6×10^6 pixels in the original image. Even with very high sampling factor such as $n/m = 4,096$, the segmentation results is still pleasing. With much less samples, the image can be clustered in high speed even without using GPU acceleration. It takes only total 0.958 seconds on CPU to perform mean shift clustering with sampling factor $n/m = 1,024$.

In Figure 4, we present image segmentation results for the images with different sizes, and give the comparison results with standard mean shift method [CM02], the accelerated method of Paris and Durand [PD07], and compactly represented KDE of Freedman and Kisilev [FK09]. We comprise with these methods on both performance and segmentation quality. Compared with [CM02], some weak features may be lost using our method since they may be incorporated into the salient features during data Gaussian KD-tree clustering, however, the salient features may be better kept, as illustrated in Figure 4. As shown in Figure 4, given the same sampling factor, our method generates higher quality compared with [PD07] and [FK09], especially at the regions with weak edges. The complexity of [PD07] depends on the dimension d of the point, when processing high-dimensional data, this method does not show much advantage. However, our method is fast even with low sampling factor and high dimensional data sets. as shown in Table 1. It takes our method 5.91 second to cluster 6.6×10^6 pixels on CPU, while it take 105.5 seconds using [PD07]. Using our method incorporating GPU implementation, our method shows greater advantage when processing large data sets with high dimensional feature vector, it takes less than 0.2 second to cluster 6.6×10^6 pixels on GPU.

4.2. Video segmentation

Mean shift clustering can also be used in video segmentation [DeM02]. As video streaming usually contains millions of pixels, practical video segmentation using mean shift clustering depends heavily on the performance of the mean shift procedure. In addition, compared with image data, the dimensions of feature space are higher, which further increase the computational complexity of mean shift procedure. Thus, it is impracticable to segment long range video streaming by performing standard mean shift clustering without using acceleration techniques. However, using our fast mean shift clustering method, we can perform video segmentation at interactive rate.

We define a feature space $\{x_i\}_{i=1}^n$ of seven dimensions for the video. The feature vector at each pixel $x_i = (\sigma_p p_i, \sigma_c c_i, \sigma_t t_i, \sigma_\varsigma \varsigma_i)$ comprises its position p_i (x and y coordinate), color c_i (Lab color vector), time t_i and motion ς_i, these four kinds of features can be weighted by parameters σ_p, σ_v, σ_t and σ_ς, and the values of these parameters are defined as constants for all pixels. As illustrated in Figure 5, there are 1.44×10^7 pixels in the video, and we first cluster the video with sampling factor 16348 using KD tree. It takes our method 16.23 seconds to perform the mean shift clustering on CPU, and 1.2 seconds on GPU. It takes 320.3 seconds on CPU using [PD07].

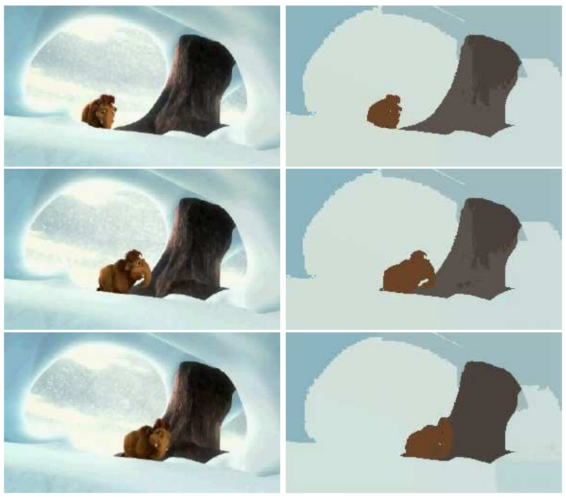
Figure 5: *Video segmentation. Left column, input video* $(600\times480\times50)$*, from top to down, 1th frame, 28th frame, 50th frame. Right column, segmentation results.*

4.3. Mesh model segmentation

Similar to image and video segmentation in image processing and analysis, surface segmentation is one of the most important operations in geometry processing and modeling. Yamauchi et al. [YLL*05] adopted the mean shift clustering to mesh partition, and produced feature sensitive mesh segmentation. In Figure 6, we give the mesh segmentation using the proposed fast mean shift cluttering. We define a feature space $\{x_i\}_{i=1}^n$ of six dimensions for the mesh model. The feature vector at vertex $x_i = (\sigma_p p_i, \sigma_v v_i)$ comprises its position p_i (x, y and z coordinate) and normal v_i (three dimension vector) which can be weighted by parameters σ_p and σ_v. The values of σ_p and σ_v are defined as global constants for all vertices.

In Figure 6, using the variant of mean shift procedure to the mesh model, we receive a patch-type segmentation results, and the segmentation results are sensitive to the salient surface features. Furthermore, we adopt the hierarchical image segmentation method [PD07] to mesh model and generate the hierarchical segmentation results. Note that our fast mean shift method is guaranteed to produce segmentation results which catch the meaningful components, no additional computation is needed to compute the hierarchical results. For a mesh model with 70,994 vertices, it takes 0.31 second for our method to compute the results. We also give the comparison results with [YLL*05]. As shown in Figure 6, our approach generates more convincing results.

4.4. Animated geometry object segmentation

The proposed fast mean shift also can be used to accelerate the animated object (geometry surface sequences) segmen-

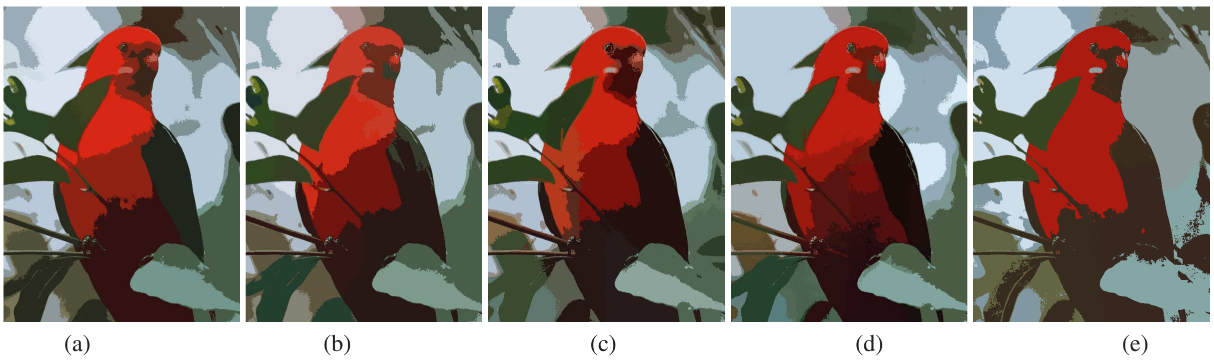

Figure 3: *Image segmentation using different sampling factor. (a) Original image, (b) n/m =256, (c) n/m =1,024, (d) n/m =4,096, (e) n/m =16,384.*

Data set	Data set size	d	n/m	Kd-tree construction	Modes computing	Modes interpolation	time
bird	564×752	5	1024	0.904	0.024	0.030	0.958
Obama	1128×1504	5	4096	1.013	0.016	0.108	1.137
castle	2256×3008	5	4096	5.105	0.088	0.717	5.910
Video	600 × 480 × 50	7	16348	13.898	0.084	2.248	16.23
Mesh	70994	6	512	0.225	0.020	0.065	0.310
Horse	30784 × 60	30	4094	1.311	0.053	0.136	1.500

Table 1: *Performance of our method for different kinds of data sets. Note that we perform clustering for animated object (Horse) on GPU. All other data sets are performed on CPU.*

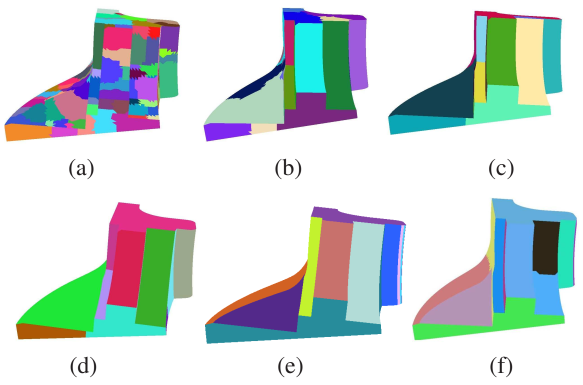

Figure 6: *Hierarchical decomposition of static mesh model. (a)-(e) is the results of our proposed hierarchical decomposition method, (f) is the result of [YLL*05].*

tation. Inspired by [LXL*], we first compute approximately invariant signature vectors ξ_i for each vertex of the animated object [LG05], which is a local and high dimensional approximately invariant under shape rigid/scaling transformations. Then both the geometric attributes (vertex position p_i and its normal v_i) and its local signature vector ξ_i of each vertex x_i on the animated object can be weighted by parameters σ_p, σ_v and σ_ξ, which construct the high dimensional feature space of the animated object $x_i = (\sigma_p p_i, \sigma_v v_i, \sigma_\xi \xi_i)$. Then the vertices of animated object can be clustered efficiently using the proposed GPU- accelerated mean shift clustering algorithm.

In Figure 7, we give the animated object segmentation results. There are total 1.6×10^6 vertices in the animated object with 50 frames. We use $d = 24$ dimensions for the signature vector ξ_i ($\xi_i \in R^{24}$) in our implementation. It takes 1.5 seconds on GPU to complete the mean shift iterations (10 iterations in this example) with sampling factor $n/m = 4096$. We also give the comparison results with [WB10]. Wuhrer and Brunton [WB10] performed the animated object segmentation in dual space of the mesh model. They found near-rigid segments whose segment boundaries locate at regions of large deformation, and assumed that the vertex-to-vertex correspondences of the input meshes were known. As an alternative, our method relies on the local high dimensional signature vector information for clustering, incorporating with the proposed fast mean shift clustering techniques, which ensures the decomposed parts more meaningful and temporally-coherent results in higher speed.

5. Conclusion

In this paper, we propose a new algorithm for accelerating compute mean shift clustering. Using KD-tree to adaptively cluster the original data set into clusters with similar feature similarity, the clusters construct the samples of the original data set. Then we compute the mean shift procedure on

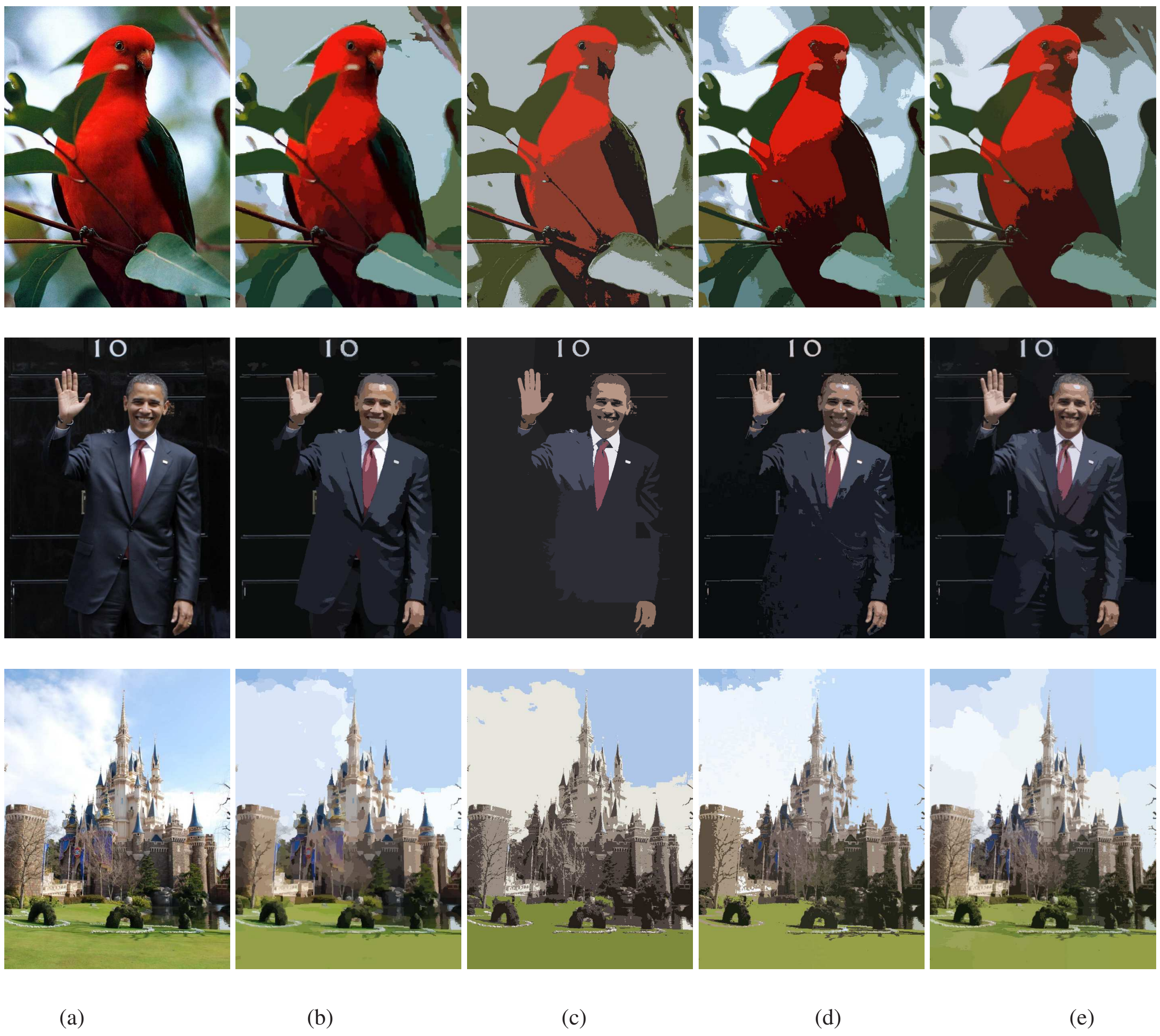

Figure 4: *Image segmentation comparisons. (a) original image, (b) image segmentation using standard mean shift (EDISON), (c) image segmentation using [PD07], (d) image segmentation using [FK09], (e) image segmentation using our proposed method.*

the greatly reduced sampled feature space and generated the modes, and finally by using the Gaussian importance weight, we upsample the computed modes to the original data set to get final clustering results. Our algorithm significantly speeds up the performance while not sacrificing the accuracy. Our method is especially useful for high resolution images, long time video sequences and geometry models segmentation with large point set.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments and insightful suggestions. This work was partly supported by NSFC (No. 60803081), National High Technology Research and Development Program of China (863 Program) (No. 2008AA121603), Natural Science Foundation of Hubei Province (2008CDB350), State Key Lab of CAD&CG (No. A0808), the Fundamental Research Funds for the Central Universities (6081005).

References

[AGDL09] ADAMS A., GELFAND N., DOLSON J., LEVOY M.: Gaussian kd-trees for fast high-dimensional filtering. *ACM Transactions on Graphics (TOG) 28*, 3 (2009), 21.

[AMN*98] ARYA S., MOUNT D., NETANYAHU N., SILVERMAN R., WU A.: An optimal algorithm for approximate near-

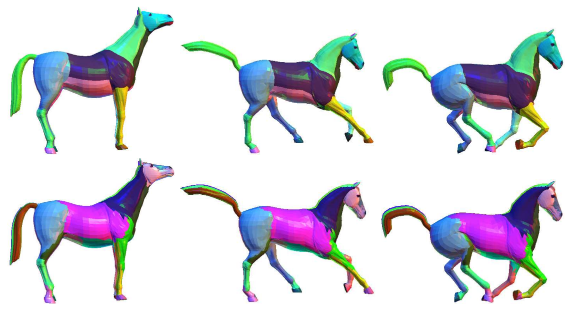

Figure 7: *Animated object decomposition result comparison. Top row: our results, bottom row: Wuhrer and Brunton [WB10].*

est neighbor searching fixed dimensions. *Journal of the ACM (JACM) 45*, 6 (1998), 891–923.

[AP08] AN X., PELLACINI F.: Appprop: all-pairs appearancespace edit propagation. *ACM Trans. Graph 27*, 3 (2008), 40.

[BC04] BARASH D., COMANICIU D.: A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. *Image and Vision Computing 22*, 1 (2004), 73–81.

[BCM05] BUADES A., COLL B., MOREL J.: A non-local algorithm for image denoising. In *CVPR 2005* (2005), pp. 60–65.

[Buc07] BUCK I.: Gpu computing: Programming a massively parallel processor. In *Proceedings of the International Symposium on Code Generation and Optimization* (2007), IEEE Computer Society, p. 17.

[Che95] CHENG Y.: Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 8 (1995), 790–799.

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence 24*, 5 (2002), 603–619.

[CP06] CARREIRA-PERPINÁN M.: Acceleration strategies for Gaussian mean-shift image segmentation. In *CVPR* (2006), vol. 1.

[CRM03] COMANICIU D., RAMESH V., MEER P.: Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 5 (2003), 564–577.

[DeM02] DEMENTHON D.: Spatio-temporal segmentation of video by hierarchical mean shift analysis. *Language 2* (2002).

[DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. In *SIGGRAPH* (2002), ACM, pp. 769–776.

[EDD03] ELGAMMAL A., DURAISWAMI R., DAVIS L.: Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 11 (2003), 1499–1504.

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 950–953.

[FH75] FUKUNAGA K., HOSTETLER L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory 21*, 1 (1975), 32–40.

[FK09] FREEDMAN D., KISILEV P.: Fast Mean Shift by compact density representation.

[GSM03] GEORGESCU B., SHIMSHONI I., MEER P.: Mean shift based clustering in high dimensions: A texture classification example. In *ICCV* (2003), pp. 456–463.

[HSA91] HANRAHAN P., SALZMAN D., AUPPERLE L.: A rapid hierarchical radiosity algorithm. *ACM SIGGRAPH Computer Graphics 25*, 4 (1991), 206.

[HSHH07] HORN D., SUGERMAN J., HOUSTON M., HANRAHAN P.: Interactive kd tree GPU raytracing. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (2007), ACM, p. 174.

[JDD03] JONES T., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics 22*, 3 (2003), 943–949.

[LG05] LI X., GUSKOV I.: Multi-scale features for approximate alignment of point-based surfaces. In *SGP* (2005), Eurographics Association, p. 217.

[LXL*] LIAO B., XIAO C., LIU M., DONG Z., PENG Q.: Fast Hierarchical Animated Object Decomposition Using Approximately Invariant Signature. *Submmited to The Visual Computer*.

[PD06] PARIS S., DURAND F.: A fast approximation of the bilateral filter using a signal processing approach. *ECCV* (2006).

[PD07] PARIS S., DURAND F.: A topological approach to hierarchical segmentation using mean shift. In *CVPR* (2007), pp. 1–8.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images.

[WB10] WUHRER S., BRUNTON A.: Segmenting animated objects into near-rigid components. *The Visual Computer 26*, 2 (2010), 147–155.

[WBC*05] WANG J., BHAT P., COLBURN R., AGRAWALA M., COHEN M.: Video cutout. *ACM Transactions on Graphics 24*, 3 (2005), 585–594.

[WLGR07] WANG P., LEE D., GRAY A., REHG J.: Fast mean shift with accurate and stable convergence. In *Workshop on Artificial Intelligence and Statistics (AISTATS)* (2007), Citeseer.

[WQ04] WEI Y., QUAN L.: Region-based progressive stereo matching. In *CVPR* (2004), vol. 1, Citeseer, pp. 106–113.

[WTXC] WANG J., THIESSON B., XU Y., COHEN M.: Image and video segmentation by anisotropic kernel mean shift. *Computer Vision-ECCV 2004*, 238–249.

[WXSC04] WANG J., XU Y., SHUM H., COHEN M.: Video tooning. In *ACM SIGGRAPH 2004 Papers* (2004), ACM, pp. 574–583.

[XLJ*09] XU K., LI Y., JU T., HU S., LIU T.: Efficient affinity-based edit propagation using KD tree. In *ACM SIGGRAPH Asia 2009 papers* (2009), ACM, pp. 1–6.

[XNT10] XIAO C., NIE Y., TANG F.: Efficient Edit Propagation Using Hierarchical Data Structure. *IEEE Transactions on Visualization and Computer Graphics* (2010).

[YDDD03] YANG C., DURAISWAMI R., DEMENTHON D., DAVIS L.: Mean-shift analysis using quasi-Newton methods. In *ICIP* (2003), vol. 3, Citeseer, pp. 447–450.

[YDGD03] YANG C., DURAISWAMI R., GUMEROV N., DAVIS L.: Improved fast gauss transform and efficient kernel density estimation. In *ICCV* (2003), pp. 664–671.

[YLL*05] YAMAUCHI H., LEE S., LEE Y., OHTAKE Y., BELYAEV A., SEIDEL H.: Feature sensitive mesh segmentation with mean shift. In *SMA* (2005), vol. 243, IEEE.

[ZHWG08] ZHOU K., HOU Q., WANG R., GUO B.: Real-time kd-tree construction on graphics hardware. In *ACM SIGGRAPH Asia 2008 papers* (2008), ACM, pp. 1–11.