IIE Transactions (2007) **39**, 581–591
Copyright © "IIE"
ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/07408170600899599

Quantile contours and multivariate density estimation for massive datasets via sequential convex hull peeling

JAMES P. McDERMOTT[1] and DENNIS K. J. LIN[2,*]

[1]*Department of Statistics and* [2]*Department of Supply Chain and Information Systems, The Pennsylvania State University, University Park, PA 16802, USA*
E-mail: DKL5@psu.edu

Received March 2004 and accepted April 2006

We propose a low-storage, single-pass, sequential method for the execution of convex hull peeling for massive datasets. The method is shown to vastly reduce the computation time required for the existing convex hull peeling algorithm from $O(n^2)$ to $O(n)$. Furthermore, the proposed method has significantly smaller storage requirements compared to the existing method. We present algorithms for low-storage, sequential computation of both the convex hull peeling multivariate median and the convex hull peeling pth depth contour, where $0 < p < 1$. We demonstrate the accuracy and reduced computation time required of the proposed method by comparing to the existing convex hull peeling method through simulation studies.

Keywords: Data depth, data mining, quantile contours, streaming data

1. Introduction

Massive datasets are becoming more and more common in modern society. They arise from sources as diverse as large call centers, internet traffic data, sales transactional records, or satellite feeds. This phenomenon presents a clear need to be able to process the data accurately and efficiently so that current analyses may be performed before becoming innundated by a continually growing store of data.

In this paper, we are specifically interested in multivariate density estimation for massive datasets. Density estimates are essential for many statistical analyses. We will utilize sets of quantile estimates to obtain an estimate of the entire cumulative distribution function (cdf) in a functional form that can then be used to obtain density estimates. We use the term "massive" to mean very large in terms of number of observations and/or number of dimensions. The sequential convex hull peeling method we present is applicable to high dimensional data. In fact, the gain in computational efficiency will increase with the number of dimensions since our method will remain linear in time.

While there are other methods available for multivariate density estimation, our method has several advantages. Parametric methods, such as the maximum likelihood, method of moments, or the parametric bootstrap all have the limitation that they must prespecify the form of the density being estimated. Nonparametric kernel density estimation is a viable alternative, but one must still specify the bandwidth parameter *a priori*. Our proposed method makes no assumptions about the form of the distribution of the data and lets the data speak for itself.

In one dimension, the concept of a *quantile* is well defined. We define the pth *quantile* as $\xi_p = F^{-1}(p) = \inf\{x : F(x) \geq p\}$, for $0 < p < 1$ (see Serfling (1980)). In more than one dimension, however, there is no universally accepted notion of a quantile. Indeed, there is not even a universal concept of ordering in more than one dimension. Hence, alternate definitions are required. Barnett (1976) and Small (1990) offer comprehensive surveys of different proposed alternative definitions of multivariate medians. One approach is to assign each point in a dataset a measure of *depth*. That is, for each point we have a measure of how far it is from some concept of the *center* of the dataset. In Liu *et al.* (1999), the authors give various examples of different definitions of data depth such as *Mahalanobis depth* (Mahalanobis, 1936), *half-space depth* (Hodges, 1955; Tukey, 1975), *convex hull peeling depth* (Barnett, 1976), *Oja depth* (Oja, 1983), *simplicial depth* (Liu, 1990), *majority depth* (Singh, 1991), and *likelihood depth*.

Of these methods, we will focus on the convex hull peeling depth method because of its ability to be adapted to the sequential low-storage form that we require. Although there are different versions of convex hull peeling (see Eddy (1982) and Green (1981) for a variation given by Tukey (1975)), we

*Corresponding author

will focus on the simple convex hull peeling version given by Barnett (1976).

Chazelle (1985) gives an optimal algorithm for determining the convex layers of a set of n planar points. However, it requires the entire dataset to be stored as this method finds the convex layer to which each point belongs. Instead, we are interested in finding the convex hull, or contour, that contains a certain percentage of the dataset. Furthermore, we require that the amount of storage used be very low relative to the entire sample size and that it be computed sequentially so that if additional data arrives, processing may be picked up where it left off without having to recompute for the entire dataset.

This paper is organized as follows. In Section 2, we introduce convex hull peeling and the computational issues associated with the existing methods. New methods for the sequential computation of both the convex hull peeling median and pth depth contour are proposed in Section 3. Simulation studies are presented in Section 4. In Section 5, we present an application of our proposed convex hull peeling algorithms to bivariate density estimation. In this section we also discuss the use of multivariate splines as a method of fitting a multivariate surface to our bivariate density estimates. In Section 6 we present an application to a real dataset from astronomy and concluding remarks are given in Section 7.

2. Convex hull peeling

The convex hull of a set of points is the minimum convex set that contains the entire dataset. In other words, it is the set of points that make up the outermost perimeter of the dataset where the polygon formed by connecting these points contains the entire dataset. Convex hull peeling is achieved by systematically identifying and then deleting the set of points that make up the convex hull of the set. For example, if we had a set of 100 points and we identified the set of, say, ten points that make up the convex hull, we would then delete these ten points from the dataset and have 90 remaining. We would now repeat the procedure by finding the set of points that make up the convex hull of the remaining 90 points and again deleting them.

To find the *convex hull peel multivariate median* we would repeat this process until we cannot peel any more. At this point we will have a hull consisting of three or more points with no points inside of it or we will have a set of either one or two points. In both cases, we cannot peel any further because we would then have no points remaining. If we have three or more points remaining, we take the centroid of the polygon formed by these points as our estimate of the multivariate median. If we have two points left, we take the average of each coordinate and use this pair of averages as the coordinates of our point estimate of the median. Obviously, if only one point is left we take that point as our estimate of the median.

We define the *convex hull peeling depth* (see Barnett (1976) and Liu *et al.* (1999)) of a point x in terms of the convex hull to which it belongs in the peeling process. Specifically, we will refer to the hull that contains approximately np observations as the *convex hull peeling* pth *depth contour* or simply as the pth *depth contour*, $0 < p < 1$, and we will denote this by C_p. Hence, to find the *pth depth contour*, we would peel the dataset consisting of n observations until approximately np observations are remaining. We say approximately here because you will rarely have exactly np points remaining at any given stage and, furthermore, np may not be an integer. Then the depth of a point x, denoted by $D(x)$, is defined as $1 - p$, where p refers to the depth contour, C_p, to which the point belongs. That is, the depth of a point is one minus the proportion of the n points that were in the original dataset that are contained by the depth contour to which the point belongs. For example, the outermost convex hull contains the entire dataset. Hence, the depth of all points that make up this hull is $1 - (n/n) = 0$. Using the above example, if there were ten points that made up this hull, they would then be discarded and the process would be repeated for the remaining 90 points. The points that made up the convex hull that contained these 90 points would all then have convex hull peeling depth of $1 - (90/100) = 0.10$. The points that make up this hull are then discarded and the process continues until the dataset can be peeled no further.

In one dimension, a depth contour reduces to a central interval of quantiles. For example, the one-dimensional depth contour of level $p = 0.5$ is just the interquartile range. That is, it is the central interval that contains 50% of the population. In two dimensions, we get a central region determined by the *shape* of the underlying distribution we are considering. For example, the standard bivariate normal distribution gives us a central circle, with center at the origin and with radius, r, uniquely determined by the level p as

$$r = \sqrt{-2\ln(1-p)}. \tag{1}$$

This result is derived as follows. We begin by considering integrating the density of the standard bivariate normal in polar coordinates, setting this equal to p, the proportion of the population within the depth contour of level p, and solving for the radius, in this case denoted by a. Hence we have

$$\begin{aligned}&\int_0^{2\pi}\int_0^a f(r,\theta) r\,\mathrm{d}r\,\mathrm{d}\theta\\ &= \int_0^{2\pi}\int_0^a \frac{1}{2\pi}\exp\left(\frac{-((r\cos\theta)^2+(r\sin\theta)^2)}{2}\right) r\,\mathrm{d}r\,\mathrm{d}\theta\\ &= 1 - e^{-a^2/2}.\end{aligned} \tag{2}$$

Now setting the result in Equation (2) equal to p and solving for a gives us the result in Equation (1).

Efron (1965) gives integral formulae for the computation of the expected number of vertices for the convex hull of a finite set of points drawn at random from a normal or a uniform distribution in two or three dimensions. For example, the expected number of vertices comprising the convex hull of a random set of points from a standard bivariate normal distribution is

$$E(V_n) = 4\sqrt{\pi}\binom{n}{2}\int_{-\infty}^{\infty} \Phi^{n-2}(p)\phi^2(p)\mathrm{d}p,$$

where Φ and ϕ are the cdf and density respectively of the standard univariate normal. The expected number of vertices grows very slowly as a function of sample size. For example, at 1000 000 the expected number of vertices is $\approx$17 and even at 1000 000 000 is only $\approx$22.

The result of this on the convex hull peeling algorithm is that, for very large datasets, it takes a very long time to peel away the outer layers because so few are peeled at each layer. For example, if we have a dataset of size 1000 000, the expected number of vertices in the convex hull of this set is 17.24 $\approx$17. Hence, after peeling away this layer we still have $\approx$ 1000 000 − 17 = 999 983 points remaining.

Similarly, for the three-dimensional normal distribution we see the same behavior (see Fig. 1). Here the expected number of vertices for a random sample of size n is given by

$$2 + 2\sqrt{3\pi}\binom{n}{3}\int_{-\infty}^{\infty} \Phi^{n-3}(p)\phi^3(p)\mathrm{d}p.$$

For a random sample of size 1000 000 drawn from this distribution, the expected number of vertices in the convex hull is 92.96 $\approx$ 93 and even when the size reaches 1000 000 000 the expected number of vertices is still only 147.77 $\approx$ 148. Thus, we see that the same problem exists in higher dimensions as well.

3. Proposed methods

There are two main problems with the existing convex hull peeling method. The first problem is that we have to physically store the entire dataset to execute the algorithm. The second problem is one of computation time. The time required to execute the existing method grows at a quadratic rate as the sample size increases. We now propose methods that address both of these issues.

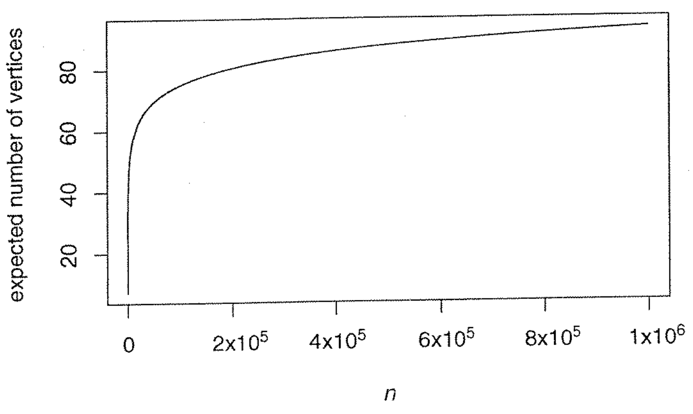

Fig. 1. Expected number of points in the convex hull of a set of points in three dimensions.

3.1. *Sequential convex hull peeling—multivariate median*

We have defined the *convex hull peeling multivariate median* as the centroid of the innermost hull obtained by successively peeling away the outermost convex hull layers. As noted above, the traditional method of obtaining this multivariate median is computationally intensive and imposes an extreme storage burden. The alternative we propose solves both of these problems by only storing a very small number of points at any one time and attacking the massive dataset with a "divide-and-conquer" strategy that yields a computational complexity that is linear in n, the sample size.

Overview of sequential convex hull peeling algorithm for estimation of the multivariate median

Step 1. Take the first m points from the dataset and peel the layers until approximately $m/2$ points are left.
Step 2. Add enough points from the remaining dataset to bring the number up to m again.
Step 3. Repeat until the dataset is exhausted.
Step 4. Peel the final set of points down to the innermost hull.
Step 5. The "center" of this resulting hull is taken as an estimate of the multivariate median.

By "center" we mean the centroid of the final hull. In practice, we take m to be between 1000 and 10 000 as this range of values has proven to work well in all situations considered. It should be noted that one could take values of m larger than 10 000, but one must weigh the improvement in performance against the increase in computation time. We discuss the implications of taking different values of m in Section 5.

3.2. *Sequential convex hull peeling—depth contours*

We have defined the *convex hull peeling pth depth contour* as the convex hull obtained by successively peeling away the outermost convex hull layers until approximately np points are within the perimeter of the hull and approximately $n(1-p)$ points are outside of the perimeter. We now present a sequential algorithm to sequentially compute and maintain the convex hull peeling depth contour for a given value of p, for $0 < p < 1$.

Overview of sequential convex hull peeling algorithm for estimation of the pth depth contour

Step 1. Take the first m points from the dataset and peel the layers until approximately mp points are left, where p represents the proportion associated with the desired quantile contour.

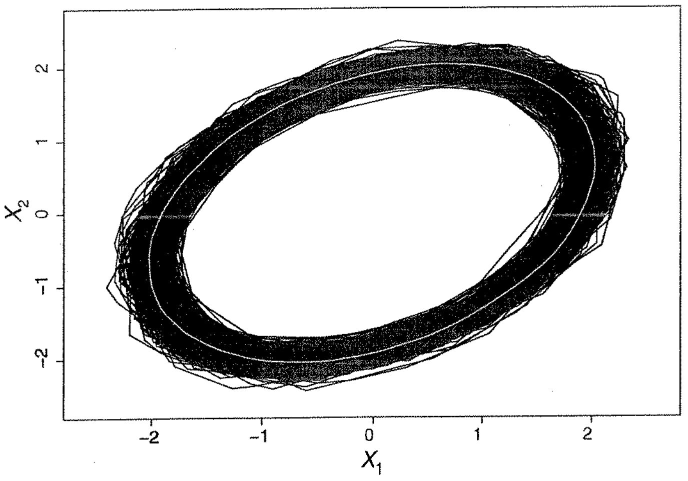

Fig. 2. Example of the quantile contour algorithm output.

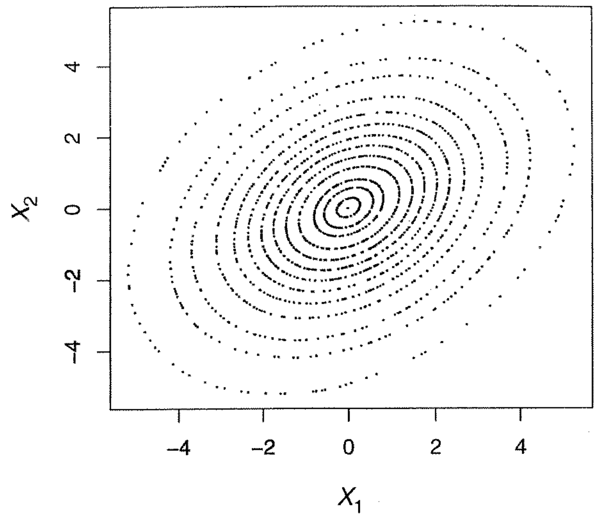

Fig. 3. Example of a set of depth contours.

Step 2. Store the b_1 points representing this convex hull.
Step 3. Repeat k times, each time appending the b_i points from the new depth contour to the previous set of points.
Step 4. Peel the resulting set of $\sum_{i=1}^{k} b_i$ points until approximately $(1/2)\sum_{i=1}^{k} b_i$ are remaining.
Step 5. This final hull is the estimate of the pth depth contour.

In Fig. 2, we give an example of the output of the sequential depth contour algorithm. The black lines are the k contours obtained from Steps 1 through 3 of the algorithm. We note that each of these black lines is really just a convex hull formed by a set of points which will then be peeled in Step 4. The white line in the center of the black lines is the depth contour obtained after completing Step 4 of the algorithm.

In Fig. 3, we give an example of a set of depth contours for a dataset of 1000 000 observations drawn from a bivariate normal distribution with a zero-mean vector and with variances of one and a covariance of three. We have used our proposed algorithms to compute the 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, and 0.99 depth contours and the convex hull peeling median. In Fig. 3, the centralmost point is the estimate of the median and the contours from the center moving out are the $p = 0.99$ through $p = 0.01$ depth contours. Again, a level of p for a given depth contour means that approximately $(1 - p) \times 100\%$ of the data is contained within it.

The decision to iteratively peel the data until approximately half the points are remaining was based on intuition: what we are looking for is a "median contour" or "centralmost contour". We peel until half of the data are remaining because the resulting contour represents the median contour of the set of pth depth contours.

4. Simulation studies

4.1. *Median simulation studies*

For the median simulation studies, we will compare the sequential convex hull peeling median to the traditional convex hull peeling median. The data for the simulations will be drawn at random from the standard bivariate normal distribution. Our comparisons will include the mean distance of each method's estimates from the origin, the variances and the ratio of the variances of the two method's estimates of the distance from the origin, and a time comparison by looking at the average time required for each method. We look at different values of m and n, the total sample size, to see if different values of these parameters will have an effect on the estimator's performance. We take $m = 1000, 2000, 5000,$ and 10 000 and $n = 100\ 000$ and 1000 000. The results of these studies are given in Table 1.

For the median studies, the performance differences between the two methods are very small. As seen in Table 1, the traditional convex hull peeling method consistently outperforms the sequential version, however, the differences are insignificant. We note that as the sample size, n, increases, the performance of both methods improves. Also, for a fixed sample size, as m, the maximum number of points stored, increases from 1000 to 10 000 the performance of the sequential version remains relatively constant. For example, in Table 1, when $n = 100\ 000$, the mean of the sequential estimates is 0.0123, 0.0122, 0.0118, and 0.0116 for $m = 1000$, 2000, 5000, and 10 000, respectively. Recall that these numbers represent the average distance of each estimator from the origin, the true center in this case.

We now consider the difference in computation time between the existing convex hull peeling median estimation method and our proposed sequential median estimation method. While changing the value of m has little or no effect

Table 1. Median study on bivariate standard normal distribution with results averaged over 100 replications

		Mean		*Variance*			*Time*	
m	*n*	*Trad*	*Seq*	*Trad*	*Seq*	*Ratio*	*Trad*	*Seq*
1000	5000	0.0411	0.0446	0.000 467	0.000 537	1.150	0.29	0.21
	10 000	0.0302	0.0330	0.000 256	0.000 302	1.177	0.81	0.46
	20 000	0.0220	0.0247	0.000 143	0.000 172	1.201	2.89	1.22
	50 000	0.0150	0.0173	0.000 063	0.000 082	1.297	11.7	2.55
	100 000	0.0105	0.0123	0.000 032	0.000 043	1.348	40.7	5.18
2000	5000	0.0413	0.0462	0.000 480	0.000 567	1.182	0.28	0.19
	10 000	0.0297	0.0324	0.000 287	0.000 310	1.078	0.78	0.47
	20 000	0.0220	0.0249	0.000 126	0.000 161	1.280	2.33	1.06
	50 000	0.0144	0.0165	0.000 065	0.000 072	1.108	10.9	2.81
	100 000	0.0106	0.0122	0.000 030	0.000 042	1.381	47.4	6.63
5000	10 000	0.0301	0.0346	0.000 261	0.000 309	1.186	1.00	0.58
	20 000	0.0224	0.0246	0.000 143	0.000 171	1.191	3.58	1.92
	50 000	0.0147	0.0164	0.000 060	0.000 071	1.192	16.8	5.47
	100 000	0.0105	0.0118	0.000 031	0.000 040	1.304	48.6	9.42
10 000	20 000	0.0219	0.0251	0.000 136	0.000 167	1.223	2.40	1.33
	50 000	0.0151	0.0164	0.000 060	0.000 067	1.124	13.2	5.50
	100 000	0.0105	0.0116	0.000 031	0.000 037	1.196	52.1	14.2

(Trad = traditional exhausted method; Seq = proposed sequential methods).

on the performance of the sequential estimator, it does have some impact on the computation time. For example, in Table 1, when $n = 100\ 000$, as m goes from 1000 to 2000 to 5000 to 10 000, the average computation time goes from 5.18 to 6.63 to 9.42 to 14.2, respectively.

In Fig. 4, we present the results of a separate time comparison study. We plot the average time taken to execute the existing method and the proposed method for the convex hull peel multivariate median for samples of differing sizes drawn from the standard bivariate normal distribution. Note that the time for the existing method grows at a quadratic rate as the sample size n increases. In contrast, the computation time required for the execution of our proposed method is linear as a function of n. Additional time comparisons of the two median estimation methods are

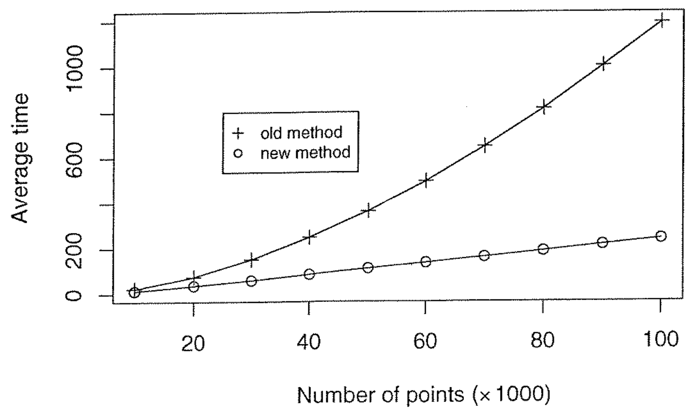

Fig. 4. Comparison of the average time taken to execute the usual convex hull algorithm and the proposed method for median estimation.

presented in Table 1 where again the same quadratic versus linear computation time is observed. For example, in Table 1, when $m = 1000$, the average computation time for $n = 5000$ is 0.29 and 0.21 for the two methods. There is not much difference at this point as the ratio of these two average times is 1.38. However, when the sample size reaches 100 000, the average computation times are 40.7 and 5.18 with the ratio of these two average times being 7.88. Based upon our experience and simulation results, we make a recommendation of setting m equal to 10 000. These settings should give slightly better accuracy and a reduction in computation time.

4.2. *Depth contour simulation studies*

For the depth contour simulation studies, we will compare the sequential convex hull depth contour to the traditional convex hull depth contour. The data for the simulations will again be drawn at random from the bivariate standard normal distribution. Since for this study we will not have point estimates as output, but rather a collection of points in the form of a convex polygon, we will use different measures to compare the method's performance.

As mentioned in Section 2, for the standard bivariate normal, the theoretical pth depth contour, C_p, will be a circle with radius $r = \sqrt{-2\ln(1-p)}$. Hence, to evaluate how closely the convex polygon, $\hat{C}_p$, determined by the set of points obtained from one of the two algorithms approximates this circle, we will convert the points output by the algorithm to polar coordinates. We will then compare how far each point is from the point on the circle with the same polar coordinate θ. That is, for a given point $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$, we calculate the angular polar coordinate for $\mathbf{x} = (x_1, x_2)$ as $\theta = \arctan(x_2/x_1)$. We then calculate the straight-line distance between $\mathbf{x} = (x_1, x_2)$ and the point on the circle, $(r\cos\theta, r\sin\theta)$. We do this for every point given as output for a given contour estimate and take the total of these errors. Since not all contours will have the same number of points, we will also take the average of these errors. Additionally, we will compare the area of the polygon, $\hat{C}_p$, with the area of the circle, C_p. This is a cross-check against the total and average error measures because if there were very few points making up the polygon, then these error measures could potentially be very small but the area most likely wouldn't be close to that of the circle, C_p. As with the median simulation study, we will also conduct an average computation time comparison. The results from this study are given in Table 2. In these studies, we examine three different contours: $p = 0.9, 0.5$, and 0.1. We also vary the value of m between 5000 and 10 000. A final variation is in the sample sizes. Table 2 considers the cases where $n = 100\,000$ and $n = 1000\,000$. As n increases, we see the similar behavior as that observed in the median simulations. Again, as with the median algorithm, we make a recommendation of setting $m = 10\,000$ for the sequential depth contour algorithm.

5. Bivariate density estimation

We now consider estimation of the bivariate density based upon the computation of expected volumes under the density surface by utilizing the depth contours discussed above. We begin with the observation that, since the pth depth contour contains $p \times 100\%$ of the population, if we could integrate the unknown density over the entire region of this depth contour it would integrate to p. That is, it would integrate to the total volume under the density and enclosed by the contour. Hence, the total volume under the surface of the density outside of the contour would integrate to $1 - p$. Using this fact, we will then solve for the heights, or values of the density, of the points in each contour by beginning at the lowest contour and "building" our way up.

As we step through this algorithm it will be helpful to refer to Fig. 5 as a reference. This figure shows a cross-section of the lower part of a density and we use it to demonstrate our density estimation procedure. Our first volume computation begins at the bottom of the density. We take the difference of the areas for the convex hull of the entire dataset, which we denote by C.inf, and the most outlying depth contour, which in our example is the 0.999th depth contour and is denoted by $C.999$. This difference gives us the total area between these two hulls, which we denote by A_1. Now, since we know the total volume under the density and *inside* of $C.999$ is 0.999, we know that the volume under the density and *outside* of $C.999$ is $1 - 0.999 = 0.001$. Hence, we will set the volume of the section under the density and outside $C.999$ equal to 0.001 and solve for the height between the two contours. We denote this first volume by V_1 and this first height by h_1. We approximate the curve of the outer surface of the density by a straight line. Note that in the cross-section in Fig. 5 the volume V_1 is represented as a triangle. For the whole volume, we must sweep this triangle around the entire circumference of $C.999$, thereby making a kind of "triangle-donut". Hence, we have that:

$$V_1 = \frac{1}{2}(\text{area}(C.\text{inf}) - \text{area}(C.999))h_1 = \frac{1}{2}A_1h_1 = 0.001,$$

and so $h_1 = 2(0.001)/A_1$. We must next compute h_2, the height from $C.999$ to $C.995$.

We now use the volume V_2 to again solve for h_2. The volume under the surface of the density and contained within $C.995$ is 0.995 and the volume outside $C.995$ is $1 - 0.995 = 0.005$. We also know the volume $V_1 = 0.001$. A further known quantity is the volume of the "rectangle-donut" beneath V_2 and beside V_1. We refer to this as a

Table 2. Contour study on bivariate standard normal distribution with results averaged over 100 replications

			Total error		*Average error*		*Area error*		*Average time*	
n	p	m	*Trad*	*Seq*	*Trad*	*Seq*	*Trad*	*Seq*	*Trad*	*Seq*
100 000	0.9	5000	0.9667	1.6247	0.0101	0.0217	7.3×10^{-3}	1.1×10^{-1}	18.00	1.31
	0.9	10 000	0.9784	1.1311	0.0101	0.0158	7.0×10^{-3}	5.3×10^{-2}	17.54	1.91
	0.5	5000	0.5811	0.7033	0.0053	0.0090	5.1×10^{-4}	5.0×10^{-3}	61.49	3.90
	0.5	10 000	0.6188	0.6504	0.0056	0.0085	6.7×10^{-4}	4.0×10^{-3}	60.59	5.85
	0.1	5000	0.3882	0.3730	0.0054	0.0073	6.8×10^{-5}	4.6×10^{-4}	80.99	5.36
	0.1	10 000	0.3757	0.3376	0.0052	0.0067	5.4×10^{-5}	2.6×10^{-4}	80.45	7.90
1000 000	0.9	5000	0.6890	2.9041	0.0034	0.0184	7.1×10^{-4}	6.6×10^{-2}	727.1	12.71
	0.9	10 000	0.6904	0.9395	0.0034	0.0068	4.7×10^{-4}	9.6×10^{-3}	702.3	17.42
	0.5	5000	0.3746	0.5713	0.0019	0.0038	3.9×10^{-5}	9.4×10^{-4}	2443.8	36.67
	0.5	10 000	0.3966	0.4279	0.0017	0.0030	3.9×10^{-5}	4.7×10^{-4}	2548.4	53.27
	0.1	5000	0.3631	0.2999	0.0024	0.0029	3.4×10^{-5}	6.9×10^{-5}	3231.0	52.03
	0.1	10 000	0.3517	0.2280	0.0023	0.0023	3.4×10^{-5}	2.6×10^{-5}	3251.4	73.47

(Trad = traditional exhausted method; Seq = proposed sequential method).

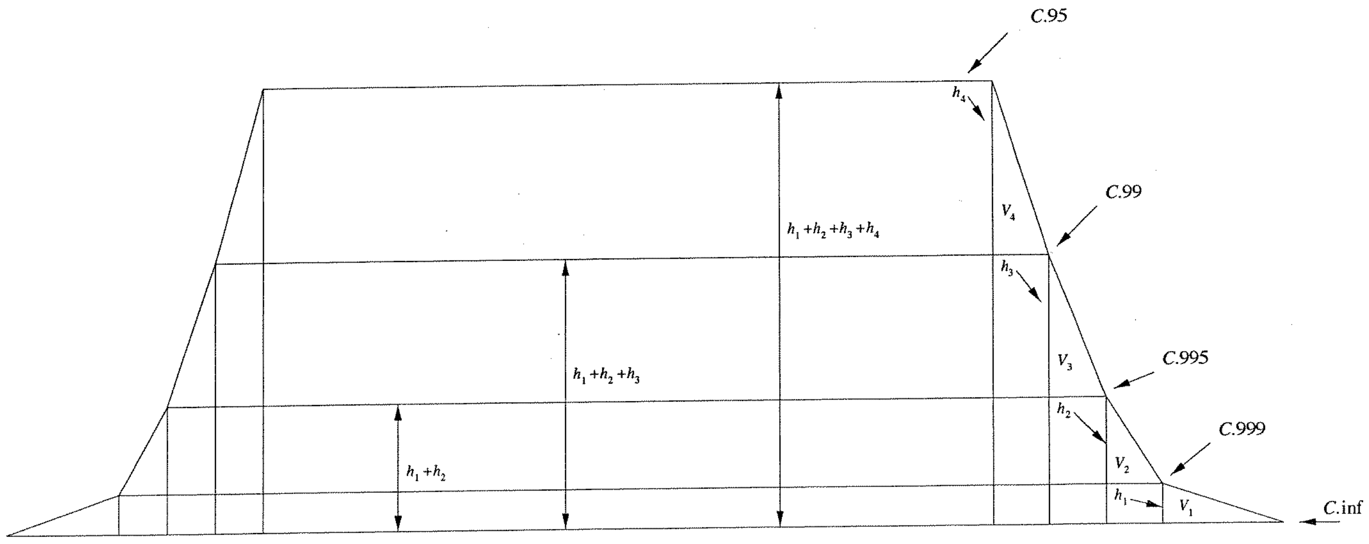

Fig. 5. Schematic for computation of the bivariate density.

"rectangle-donut" because we sweep the rectangle around the circumference of $C.999$. The volume of this piece is just $(C.999 - C.995)h_1$, that is, it is the product of the difference in the areas of $C.999$ and $C.995$ and the just computed height, h_1. Hence, we have that:

$$\begin{aligned}0.005 &= 0.001 + V_2 + (\text{area}(C.999) - \text{area}(C.995))h_1\\ &= 0.001 + \frac{1}{2}(\text{area}(C.999) - \text{area}(C.995))h_2\\ &\quad + (\text{area}(C.999) - \text{area}(C.995))h_1\\ &= 0.001 + \frac{1}{2}A_2h_2 + A_2h_1,\end{aligned}$$

and so $h_2 = 2(0.004 - A_2h_1)/A_2$.

Continuing in this manner, we may determine all of the heights, h_i. Then to obtain the density estimate for a given contour, we add up the heights until we reach the desired contour. For example, in Fig. 5, to obtain the density estimate for points in the $C.995$ contour, we add the first two heights. So, $h_1 + h_2$ would be our density estimate for all points that make up this depth contour.

We now generalize this procedure. Assume we have $0 < p_k < \cdots < p_2 < p_1 < 1$ and corresponding depth contours, $C_{p_k}, \ldots, C_{p_2}, C_{p_1}$, and define C_{inf} to be the convex hull of the entire dataset. Let $A_1 = \text{area}(C_{inf}) - \text{area}(C_{p_1})$ and $A_i = \text{area}(C_{p_{i-1}}) - \text{area}(C_{p_i})$, for $i = 2, \ldots, k$. Then we have

$$h_1 = \frac{2(1-p_1)}{A_1},$$
$$h_2 = \frac{2((p_1-p_2) - A_2h_1)}{A_2},$$

and in general

$$h_i = \frac{2((p_{i-1}-p_i) - A_i(h_1 + \cdots + h_{i-1}))}{A_i},$$

for $i = 3, \ldots, k$.

In Table 3, we present the results of a simulation study to quantify the performance of our proposed bivariate density estimator. We have generated 1000 000 observations from the bivariate standard normal distribution and computed the 0.999, 0.995, 0.99, 0.95, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.05, 0.01, 0.005, and 0.001 depth contours. Recall that these contours as listed go from outermost to innermost since the 0.999th contour contains 99.9% of the data and the 0.001th depth contour contains 0.1% of the data. We ran the simulation for $m = 10\,000$ and $m = 20\,000$ and averaged results over 100 iterations. We then compare the averaged estimates to the true density value for the points

Table 3. Density estimation for the bivariate standard normal density with $n = 1000000$ and results averaged over 100 replications

		m = 10 000	
p	*True density*	*Mean*	*Variance*
0.999	0.000 16	0.000 09	5.7×10^{-11}
0.995	0.000 80	0.000 64	6.3×10^{-10}
0.99	0.001 59	0.001 63	6.1×10^{-09}
0.95	0.007 96	0.006 43	1.5×10^{-08}
0.90	0.015 92	0.016 80	5.8×10^{-08}
0.80	0.031 83	0.029 32	1.3×10^{-07}
0.70	0.047 75	0.049 43	4.0×10^{-07}
0.60	0.063 66	0.061 47	1.2×10^{-06}
0.50	0.079 58	0.081 60	2.6×10^{-06}
0.40	0.095 49	0.093 20	4.9×10^{-06}
0.30	0.111 41	0.114 04	7.2×10^{-06}
0.20	0.127 32	0.124 95	8.5×10^{-06}
0.10	0.143 24	0.147 20	1.1×10^{-05}
0.05	0.151 20	0.147 55	1.8×10^{-05}
0.01	0.157 56	0.156 96	2.4×10^{-05}
0.005	0.158 36	0.159 35	1.3×10^{-04}
0.001	0.159 00	0.166 23	2.7×10^{-04}

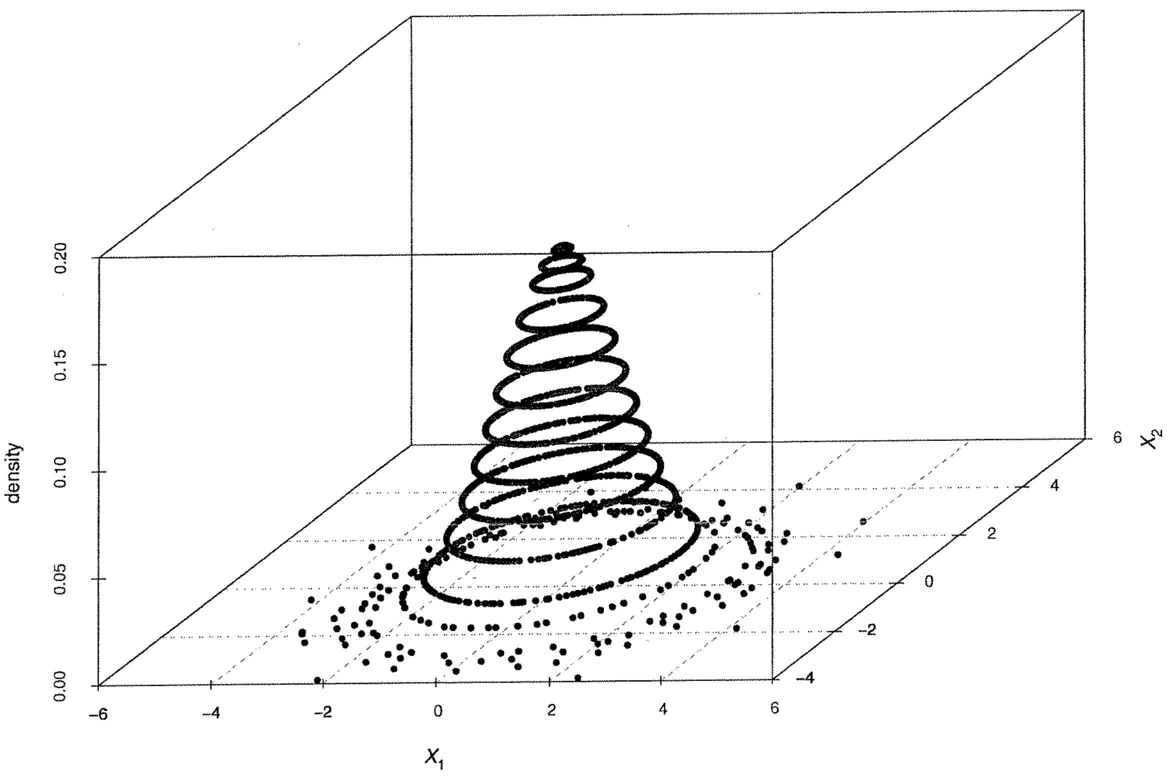

Fig. 6. Three-dimensional contours of a standard bivariate normal density.

in each contour and we also examine the variance of the 100 estimates.

One can see that the results are quite accurate. For example, with $m = 10\,000$, the 0.001th contour averaged estimate was 0.00 009 compared to the true density value of 0.000 16, while the 0.999th contour averaged estimate was 0.16 623 compared to a true density value of 0.159 00. We also note that increasing m from 10 000 to 20 000 made no appreciable difference in performance, but this increase did cause the running time to increase dramatically. This gives empirical support for our recommendation of $m = 10\,000$.

Having thus obtained density estimates for all of our depth contours, we may then wish to fit a multivariate spline to our results. Refering back to Fig. 3, we see an example of a set of contours obtained from a sample of 1000 000 observations taken from the standard bivariate normal distribution. We then assign heights, obtained by the above bivariate density estimation method, to these two-dimensional contours. We plot the resulting three-dimensional points in Fig. 6, and in Fig. 7 we present the result of a multivariate spline fit to the three-dimensional set of points made up by the points in the contours and their associated estimated density values.

6. Application to an astronomy dataset

We now apply our methods to an astronomy dataset as an illustration. The data is taken from the Digital Palomar Observatory Sky Survey (DPOSS). We have a set of 11 355 objects in the sky and we will be looking at measures of the magnitude of these sky objects in the blue-green and near-infrared visible bands. Although the dataset we consider has only 11 355 objects, the DPOSS project projects

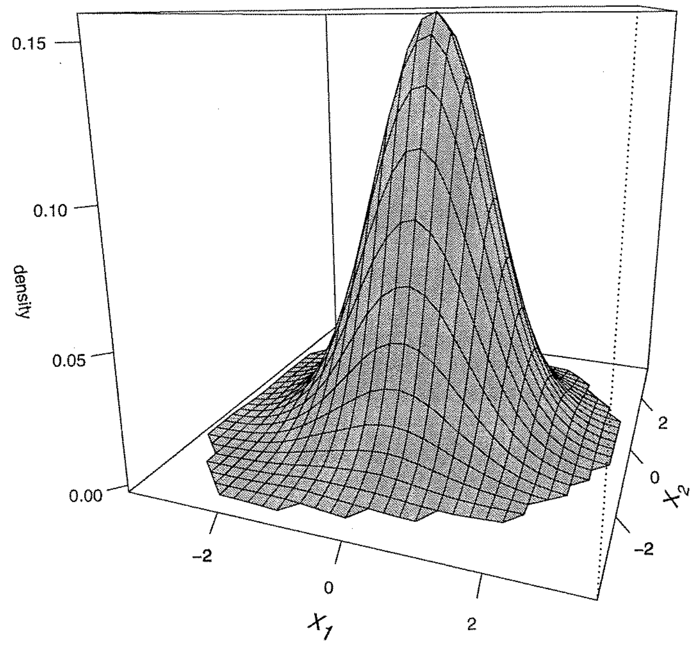

Fig. 7. Example of multivariate spline fit to contours.

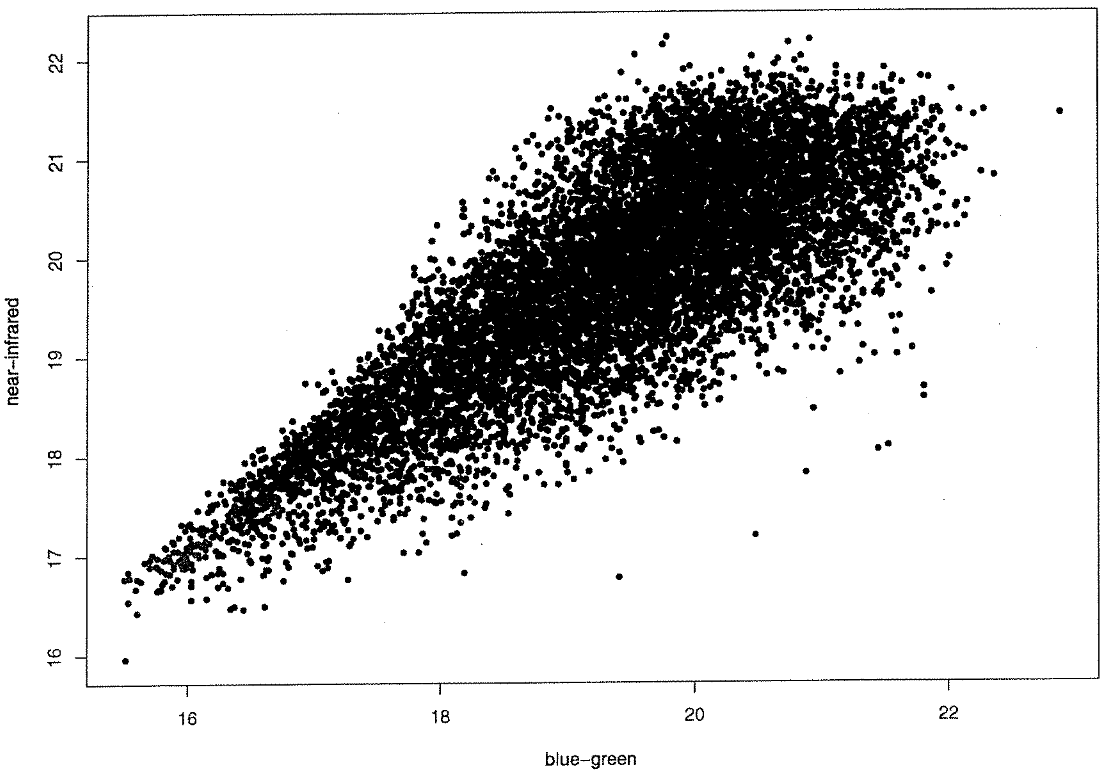

Fig. 8. DPOSS dataset used in example with $n = 11\,355$.

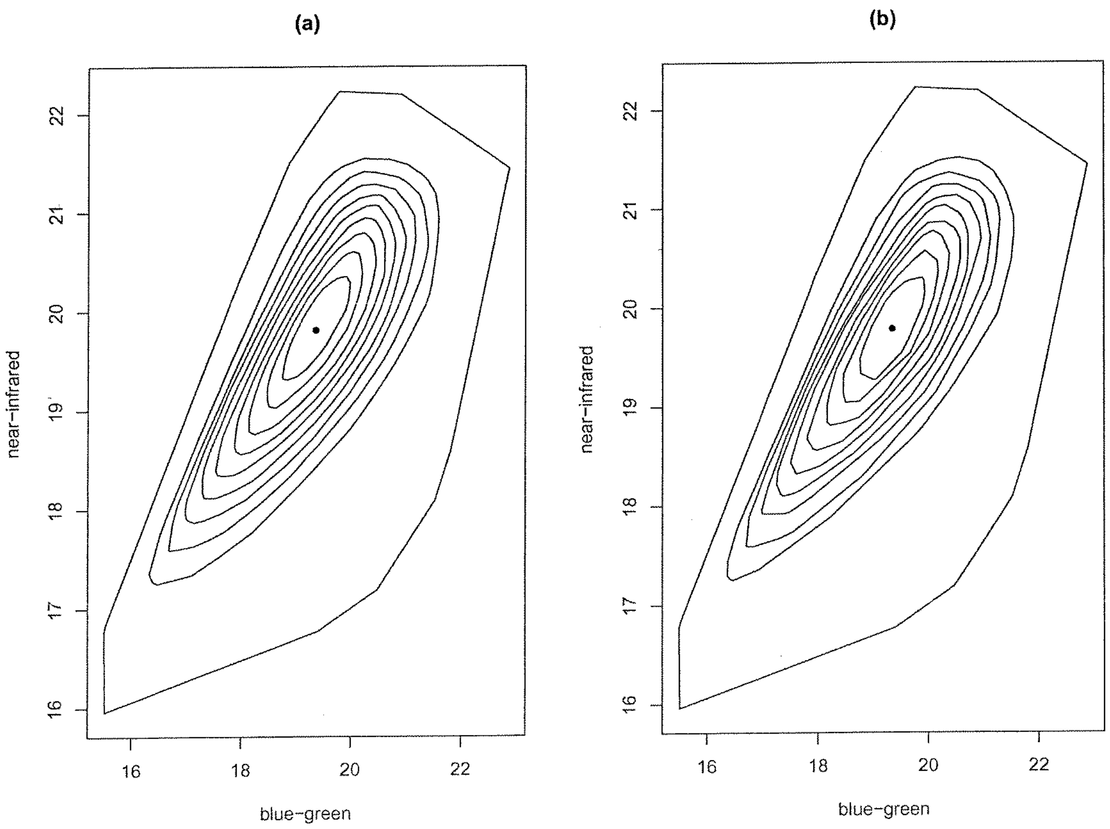

Fig. 9. Convex hull peeling depth contours and medians using: (a) traditional methods; and (b) sequential methods.

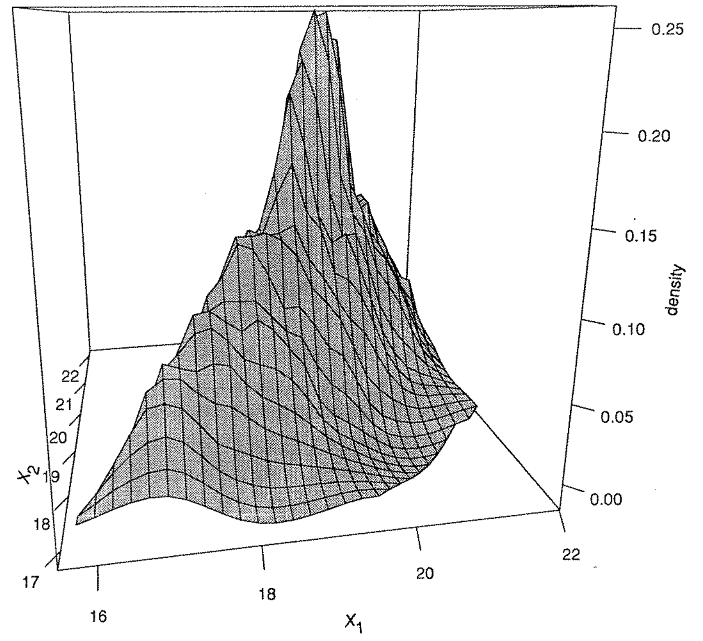

Fig. 10. Multivariate spline fit to the depth contours with density estimates from the DPOSS dataset.

that over 50 000 000 galaxies and over two billion stars will be contained in the final sky catalog (Djorgovski *et al.*, 2002).

In Fig. 8, we present a scatterplot of the DPOSS dataset used in this example. In Fig. 9(a and b), we give the results of executing both the traditional and sequential convex hull peeling algorithms on the DPOSS dataset as a side-by-side graphical comparison. We compute the multivariate medians, which are represented by a central point, and the sets of depth contours ranging from the centralmost contour with $p = 0.1$ out to the farthest contour with $p = 0.9$. Despite utilizing only 11 355 data points, visually there is not much difference between the two graphs. The outermost contour in Fig. 9(a and b) is the convex hull of the entire dataset. This is given as a visual reference. We note here that the outermost convex hull of the entire dataset may easily be computed sequentially by taking m points at a time, taking the convex hull of these points, and then taking the convex hull of the union of this hull and the next m points. Continuing in this manner we may keep a sequential version of the outermost hull and this will be exactly the same as if we had stored the entire dataset and taken the convex hull. We quantify the difference between the areas of depth contours for the two methods as follows from the outermost contour to the innermost: 0.6603, 0.3690, 0.3364, 0.2541, 0.0791, 0.0736, 0.0488, 0.0128, and 0.0593. The distance between the traditional and sequential multivariate medians was 0.0258. In Fig. 10, we fit a multivariate spline to the contours using the density estimation method given in Section 5.

7. Concluding remarks

One drawback to the sequential depth contour algorithm is that, although it uses much less storage than the traditional method, it still requires perhaps more storage than we would like. Experiments have shown that approximately 1–2% of the dataset must be stored to execute the algorithm. There is room for improvement in this area. Another thing to note is that at this time the depth contours must be computed separately. Hence, to avoid having to make more than one pass through the data, one would have to program the algorithm in parallel with each individual contour being computed by a separate processor.

Another potential drawback to this method is that there is very little theory available for convex hulls. Hence, we do not at this time have a good sense of breakdown properties, i.e., the robustness of the estimator. A further drawback is that standard errors are not available for our estimates and hence to develop inferential methods for our estimators, we would need to develop other methodologies to get some measure of the variability of the estimates. We have considered the possibility of various resampling methods, such as the bootstrap, but this is an area for future work.

By utilizing the contours and recognizing their relationship to expected volumes under the surface of the unknown density, we are able to obtain bivariate density estimates for all points on each contour without making any choice of bandwidth as we must do with kernel density estimation. We may then fit a multivariate spline to the resulting three-dimensional set of points to obtain the functional form of the entire density. The ideas in this paper can be extended to dimensions higher than two. This is an area of future work.

The linear complexity of the proposed algorithms occurs because at any given time, we have at most m points in local storage for these methods. Each set of m points will have the same algorithm steps applied to it, with each step having a maximum amount of time needed for its execution. Hence, as the size of the dataset grows, the time needed to execute the algorithm will continue to grow at a linear rate.

Clearly there could be situations where, if the data was not presented in a "random" manner, the proposed algorithms could have problems. However, these methods should be able to "adapt" to minor changes in the underlying distribution of the data as it arrives. The sequential nature of the methods allows for distributional changes over time. For example, if the "center" of the distribution was drifting over time, the median estimation method should be able to drift with it. This is another area for future work.

The convex hull of a random set of points in two dimensions is a convex polygon. In three dimensions, the convex hull will be a *polyhedron*, a region in three-dimensional space whose boundary is made up of a finite number of two-dimensional polygon *faces*. Any two of these faces will be disjoint and meet at *edges* and *vertices*. O'Rourke (1998) gives algorithms for the computation of the convex hull in three dimensions. The *quickhull* algorithm (see Barber *et al.* (1996)) can be used to obtain the convex hulls for datasets of arbitrary dimension. Although this method can handle arbitrarily large dimensional data, this method will face the same problem as other methods in terms of computation time. Hence, our approach can be used in conjunction with the quickhull algorithm when the data is in the form of a stream with no fixed length.

The code for these methods was written exclusively with the R programming language as the basic algorithm for finding the convex hull of a planar set of points is part of the base R code. For a description of the algorithm used by the R progamming language, see Eddy (1977a, 1977b).

Acknowledgements

The authors wish to thank the referees for their comments as they have greatly improved the quality of the paper. Dennis Lin is partially supported by a Research Grant Competition, Center for Supply Chain Research, The Smeal College at Penn State University.

References

Barber, B.C., Dobkin, D.P. and Huhdanpaa, H. (1996) The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, **22**(4), 469–483.

Barnett, V. (1976) The ordering of multivariate data. *Journal of the Royal Statistical Society A*, **139**(3), 318–344.

Chazelle, B. (1985) On the convex layers of a planar set. *IEEE Transactions on Information Theory*, **31**, 509–517.

Donoho, D.L. and Gasko, M. (1992) Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *Annals of Statistics*, **20**(4), 1803–1827.

Eddy, W.F. (1977a) A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software*, **3**(4), 398–403.

Eddy W.F. (1977b) Algorithm 523: CONVEX, a new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software*, **3**(4), 411–412.

Eddy, W.F. (1982) Convex Hull Peeling, Compstat 82–Part 1. *Proceedings in Computational Statistics*, Physical-Verlag, Vienna, 42–47.

Efron, B. (1965) The convex hull of a random set of points, *Biometrika*, **52** (3/4), 331–343.

Green, P.J. (1981) Peeling bivariate data, in *Interpreting Multivariate Data*, Barnett, V., (ed.), New York, NY, pp. 3–20.

Hodges, J.L. (1955) A bivariate sign test. *Annals of Mathematical Statistics*, **20**, 523–527.

Liu, R.Y. (1990) On a notion of data depth based on random simplices. *Annals of Statistics*, **18**, 405–414.

Liu, R.Y., Parelius, J.M. and Singh, K. (1999) Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Annals of Statistics*, **27**(3), 783–858.

Mahalanobis, P.C. (1936) On the generalized distance in statistics. *Proceedings of the National Academy of Sciences*, India, **12**, 49–55.

Oja, H. (1983) Descriptive statistics for multivariate distributions. *Statistics and Probability Letters*, **1**, 327–332.

O'Rourke, J. (1998) *Computational Geometry in C*, Cambridge University Press, Cambridge, UK.

Serfling, R.J. (1980) *Approximation Theorems of Mathematical Statistics*, Wiley, New York, NY.

Singh, K. (1991) A notion of majority depth. Technical Report, Rutgers University.

Small, C.G. (1990) A survey of multidimensional medians. *International Statistical Review*, **58**(3), 263–277.

Tukey, J. (1975) Mathematics and picturing data, in *Proceedings of the 1975 International Congress of Mathematics*, Vancouver, B.C., Vol. 2, pp. 523–531.

Biographies

James P. McDermott is a Senior Research Biostatistician working in the Nonclinical Biostatistics Department of the Global Biometric Sciences Division at Bristol-Myers Squibb, Wallingford, CT. He received his Ph.D. in Statistics from the Pennsylvania State University.

Dennis K.J. Lin is a University Distinguished Professor of Supply Chain and Statistics at the Pennsylvania State University, University Park. He received a Ph.D. in Statistics from the University of Wisconsin at Madison. He is an elected Fellow of the American Statistical Association, an elected member of the International Statistical Institute and a senior member of the American Society for Quality. He has published over 100 papers in the areas of design of experiments, response surface methodology, reliability, control charts, data mining and statistical inference.