Fast & Accurate Gaussian Kernel Density Estimation

Unknown Author*

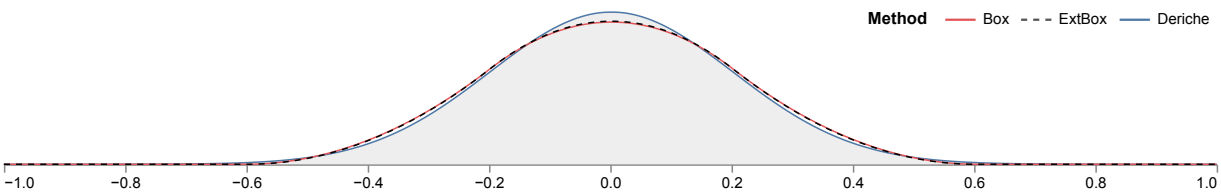

Figure 1: Gaussian kernel density estimation for a single impulse value (m = 512 bins, σ = 0.2). Iterated uniform ("box") filters [8, 26] (red & dashed) underestimate the mode and overestimate the sides of the distribution. Deriche's [4, 5] linear-time recursive filter approximation (blue) produces a pixel-perfect match to the true distribution (grey).

Abstract

Kernel density estimation (KDE) models a discrete sample of data as a continuous distribution, supporting the construction of visualizations such as violin plots, heatmaps, and contour plots. This paper draws on the statistics and image processing literature to survey efficient and scalable density estimation techniques for the common case of Gaussian kernel functions. We evaluate the accuracy and running time of these methods across multiple visualization contexts and find that the combination of linear binning and a recursive filter approximation by Deriche efficiently produces pixel-perfect estimates across a compelling range of kernel bandwidths.

1 Introduction

Kernel density estimation (*KDE*) [14, 17] estimates a continuous probability density function for a finite sample of data. KDE is regularly used to visualize univariate distributions for exploratory analysis in the form of area charts or violin plots [3, 9], providing valuable alternatives to histograms. In two dimensions, KDE estimates produce smoothed heatmaps that can be visualized directly as images or used to extract density isolines [13] for contour plots.

To form a density estimate, each data point is modeled as a probability distribution, or *kernel*, centered at that point. The kernel is parameterized by a *bandwidth* σ that determines the width (or spread) of each point distribution. The sum of these kernels constitutes the density estimate for the sample. While a variety of kernel functions exist, the normal (Gaussian) distribution is a common choice [20], in which case the bandwidth σ is its standard deviation.

We would like density estimation to be *fast*: scalable to large datasets, yet amenable to interactive bandwidth adjustment. A naïve calculation has quadratic $O(m*n)$ complexity: we must sum the contributions of n data points at each of m locations at which we measure the density. While approximation methods exist, we want them to be *accurate*, as inaccurate estimates can result in visualizations with missing features or false local extrema (peaks or valleys).

This paper reviews scalable, linear-time approximations of Gaussian kernel densities that smooth a binned grid of values. We evaluate a set of methods – *box filters* [25], *extended box filters* [8], and *Deriche's approximation* [4, 5] – in the context of 1D area charts and 2D heatmaps. We find that the combination of linear binning (proportionally dividing the weight of a point among adjacent bins) and Deriche's approximation is both fast and highly accurate, outperforming methods currently used in existing visualization tools and often providing pixel-perfect results.

*e-mail:

2 Density Estimation Methods

Given a dataset with n data points $x_i \in \Re$, a kernel function K, and bandwidth σ, the univariate kernel density estimate is defined as:

$$f(x) = \frac{1}{n\sigma} \sum_{i=1}^{n} K\left(\frac{x - x_i}{\sigma}\right) \tag{1}$$

We focus on the case where K is the normal (Gaussian) density $K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. We can directly calculate the density at a point x by summing the kernel response for each data point. If we query the density at m measurement points, this approach has computational complexity $O(m*n)$, which for large datasets can be untenable.

Nevertheless, direct calculation is used by multiple tools. At the time of writing, Vega and Vega-Lite [18, 19] use direct calculation for one-dimensional KDE, where m is the number of points queried in order to draw the density. Line segments then connect these measured values. The `kde2d` function of R's MASS [24] package (invoked by the popular ggplot2 [27] library for 2D density estimates) also uses a direct calculation approach, limiting the feasible number of measurement points for plotting. One can optimize direct calculation by leveraging spatial indices (e.g., KD trees or Ball trees) to approximate the contribution of suitably distant points [7], as done in the Python scikit-learn library [15]. However, the asymptotic complexity remains a superlinear function of n.

To speed estimation, statisticians proposed *binned KDE* methods [21] that first aggregate input data into a uniform grid with m bins. KDE then reduces to the signal processing task of smoothing the binned data. For example, one can directly convolve the binned values with a discrete Gaussian kernel (or *filter*). The resulting complexity $O(n+m*w)$ is dependent not just on the number of bins m, but also the filter width w. Larger bandwidths can result in filter widths on the same order as m, for a quadratic running time.

Silverman [23] instead applies the Fast Fourier Transform (FFT), an approach used by R's `density` routine. A strength of this method is that it can support arbitrary kernel functions: the binned data and a discretized kernel response are separately mapped to the frequency domain using the FFT, the results are multiplied element-wise (convolution in the frequency domain), and an inverse FFT then produces the density estimate. The complexity is $O(n+m\log m)$, with binning of n points followed by FFT calls on m-sized grids.

For even faster estimates, linear-time $O(n+m)$ approximations exist. These are particularly attractive for 2D density estimation, which can be computed using a series of 1D convolutions along every row and every column of a binned 2D grid. One can approximate 1D Gaussian convolution by iteratively applying a filter of uniform weight, also known as a **box filter**. Wells [26] applies this method to density estimation, contributing a formula for the filter length w (or equivalently, its radius r) as a function of both σ and the number of filter iterations k. An attractive property of this approach is its

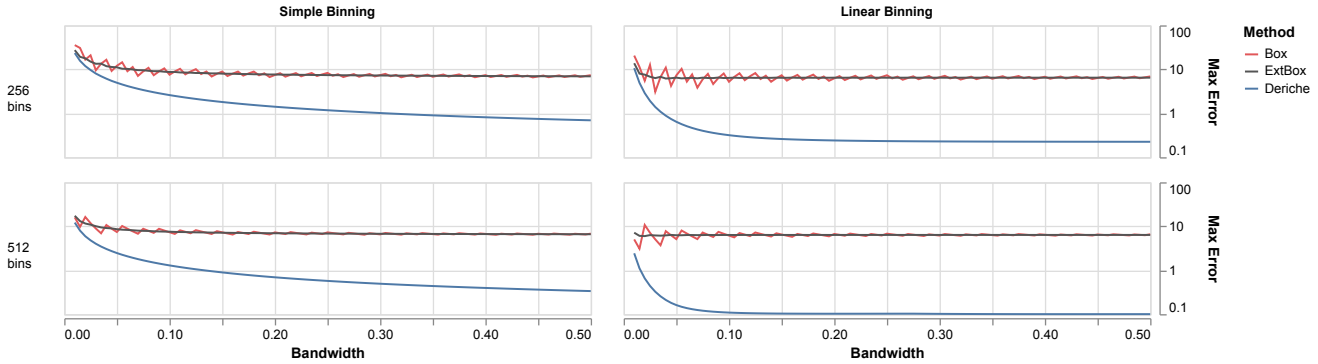

Figure 2: 1D estimation error for a single impulse. Error (plotted on a log scale) is measured as the maximum pixel error given a 100-pixel plot height. Box filters exhibit an oscillating pattern due to bandwidth quantization; the extended box method smooths these artifacts. Deriche approximation consistently produces lower error, typically with sub-pixel accuracy. Linear binning further reduces the error rate.

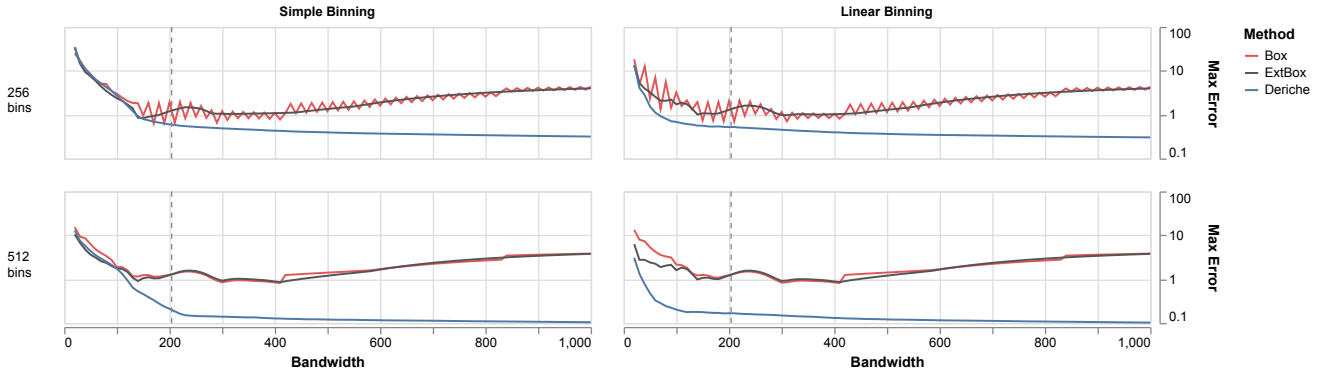

Figure 3: 1D estimation error for Gentoo penguin body mass. Error (plotted on a log scale) is measured as the maximum pixel error given a 100-pixel plot height. Dashed gray lines indicate the normal reference density (NRD) heuristic for automatic bandwidth (σ) selection [20]. The combination of linear binning and Deriche approximation consistently produces the most accurate estimates.

simplicity of calculation: k iterations of uniform filtering (running sums), followed by a scale adjustment. Both d3-contour [1] and Vega use per-row and per-column box filters for 2D density estimation.

Box filtering runs in linear-time, but has important nuances. First, the bandwidth σ (a continuous value) is discretized to a filter with integer radius r, leading to quantization error. To address this issue, Gwosdek et al. [8] propose an **extended box filter** that introduces some non-uniformity by adding fractional weight to the endpoints of the filter. Second, the true grid size is no longer a constant parameter such as $m = 512$ bins. As iterated filters 'blur' weight into adjacent bins, the grid must be extended on both ends by an offset of $k*r$. The running time scales as $n+l$, where $l = m+2k*r$. As the filter radius r is a monotone function of σ [26], this can result in larger grids – and thus higher time and memory costs – for larger bandwidths.

Finally, we consider an approximation developed by **Deriche** [4, 5] for computer vision applications. Deriche models the right half of the standard Gaussian using an order-K approximation:

$$h_K(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=1}^{K} \alpha_k e^{-x\lambda_k/\sigma} \tag{2}$$

From this formulation, Deriche constructs a recursive filter that proceeds linearly from the first value to the last. The left half of the Gaussian is defined by reversing the equation and subtracting the sample at $x = 0$ (so that it is not included twice) to form a second filter. We use a fourth-order approximation ($K = 4$), with coefficients

$$\alpha_1 = 0.84 + i1.8675, \qquad \alpha_3 = -0.34015 - i0.1299$$
$$\lambda_1 = 1.783 + i0.6318, \qquad \lambda_3 = 1.723 + i1.997$$

and $\alpha_{2k} = \alpha^*_{2k-1}$, $\lambda_{2k} = \lambda^*_{2k-1}$, where x^* denotes the complex conjugate. Deriche determined the α_k and λ_k parameters using numerical optimization to find the ℓ^2-best fit to the Gaussian over the domain $n = 0, \ldots, 1000$ with $\sigma = 100$. Getreuer [6] describes how to algebraically rearrange the terms of these filters into direct summations.

After a constant time initialization to compute summation terms for a chosen σ, the algorithm requires a linear pass over the m bins for each filter, plus a final pass to sum their results. To handle boundary values, the algorithm must in general perform iterative initialization per filter, requiring at most another linear pass. Fortunately, this initialization reduces to a constant time operation for zero-padded data (i.e., where no weight resides outside the bins).[1] Deriche's method has complexity $O(n+m)$; it involves more arithmetic operations per step than box filters, but does not require padding the binned grid. As we will see, it is also much more accurate. To the best of our knowledge, this work is the first to apply Deriche's approximation to the task of kernel density estimation.

While other methods for approximating Gaussian convolution have been proposed, they exhibit higher error rates and/or longer running times than those above. For more details, see Getreuer's survey and evaluation in the context of image filtering [6].

3 Binning Schemes

For binned KDE approaches one must choose how to bin input data into a uniform grid. By default we assume the weight of a data point is 1; however, a binned grid can easily accommodate variably-weighted points. **Simple binning**, commonly performed for histograms, places all the weight for a data point into the single bin interval that contains the point. **Linear binning** [11, 25] is an alternative that proportionally distributes the weight of a point between adjacent bins. If a data point x_i lies between bins with

[1] In contrast to zero-padded data, one must perform iterations for reflected signals (used in image processing to blur without decreasing image brightness) or periodic domains (where the final bin wraps back to the first).

midpoints b_0 and b_1, linear binning assigns weight proportional to $(b_1 - x_i)/(b_1 - b_0)$ to bin b_0 and $(x_i - b_0)/(b_1 - b_0)$ to bin b_1. For example when a point lies at the exact center of a bin, that bin receives all the weight. If a point resides near the boundary of two bins, its weight is split nearly evenly between them. We examine both binning approaches in our evaluation below.

4 Evaluation

How well do the above estimation methods perform in practice? Getreuer [6] evaluates a suite of Gaussian filtering methods in the context of image processing (e.g., blurring), inspiring the choice of methods we evaluate here. That survey does not address the question of binning method (images are already discretized into pixels) and assumes reflected signals outside the image boundaries (for filters that preserve overall image brightness). Bullmann et al. [2] examine approximate KDE methods for sensor fusion applications. They do not evaluate visualization directly, and only assess box filter approaches, omitting alternative methods such as Deriche approximation.

We evaluate KDE methods in a visualization context, assessing box filters, extended box filters, and Deriche approximation. For the box filter methods, we use $k = 3$ iterations of filtering. Using 4 or 5 iterations trades-off longer running times for higher accuracy, but the results remain qualitatively similar. For the Deriche method, we use a 4th-order recursive filter approximation [6]. Datasets and benchmark scripts are included as supplemental material.

4.1 Method

We compare the speed and accuracy of KDE methods for both univariate and bivariate visualizations. To measure accuracy, we compare against a direct calculation approach. As pixel-based visualizations are inherently discrete, we compute 'ground truth' density estimates at the per-pixel level. We treat each pixel as a bin and calculate the probability mass it contains, which is the difference in the KDE cumulative distribution function between the ending and starting bin boundaries. We then compare these values to estimates from approximation methods. For the approximate methods, we linearly interpolate results across the m bins to produce pixel-level estimates; this matches the standard plotting method of connecting density measurement points with line segments.

To evaluate accuracy in a visualization context, we consider how density plots are presented. It is common to use a linear scale with a domain that ranges from zero to the maximum density. To mirror this, prior to comparison we separately scale the density estimates, dividing each by its maximum value. We then multiply by a factor of 100, so that estimates lie on a [0, 100] scale. This provides an interpretable measure of error: discrepancies between methods correspond to the number of pixels difference in a 100 pixel-tall chart (a reasonable size for showing distributions, including violin plots), and simultaneously conveys a percentage difference.

We report the maximum (L_∞) error, indicating the most 'glaring' mistake a method makes; root-mean-square error gives qualitatively similar results. For 1D estimation we compare to ground truth estimates for a 1024 pixel wide chart. For 2D estimation we compare to ground truth for a 512×512 heatmap. Both were chosen to align with common resolutions and computation constraints.

Automatic bandwidth selection for kernel density estimation has received a great deal of attention [20, 22]. To contextualize our results, we use Scott's normal reference density (NRD) rule [20], a common default intended to minimize the asymptotic mean integrated squared error (MISE) relative to a normal distribution.

Each method was implemented as a single-threaded routine in JavaScript for web-based visualization. Benchmarks were run in Node.js v15.12.0 on a 2017 MacBook Pro with a 2.9 GHz Intel Core i7 processor. We used the `performance.now` method of the `perf_hooks` package to measure running times over repeated runs.

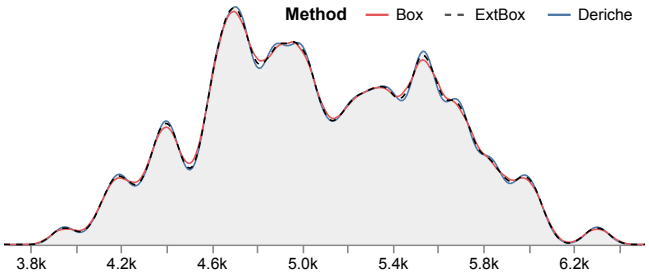

Figure 4: KDE of Gentoo penguin body mass (m = 512 bins, σ = 50). Box filters tend to underestimate peaks and overestimate valleys, in some cases 'eroding' local peaks (e.g., around 4.9k & 5.7k grams). Deriche approximation instead produces a pixel-perfect result.

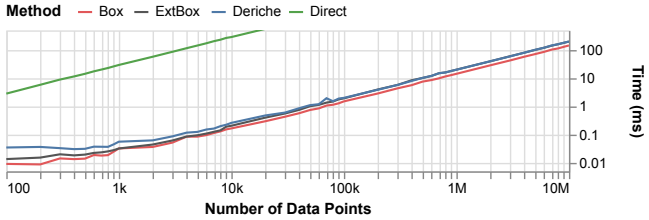

Figure 5: Running time of 1D estimation on resampled penguin data (m = 512 bins). As n increases, the running time of the approximation methods is dominated by the $O(n)$ binning cost.

4.2 Results: 1D Estimation

We first evaluate the KDE methods relative to an impulse: we locate a single point at $x = 0$ and perform estimation over the domain $[-1, 1]$. The result should be a Gaussian distribution with mean 0 and standard deviation matching the kernel bandwidth σ. Figure 1 shows the result for $\sigma = 0.2$ and $m = 512$ bins (sans re-scaling). The box filter methods produce perceptible errors, whereas Deriche approximation provides a pixel-perfect match to the actual distribution.

Figure 2 presents scaled comparisons by binning scheme, bin count, and bandwidth. Standard box filters produce oscillating errors due to bandwidth quantization. The extended box method smooths these artifacts. Deriche approximation consistently produces the lowest error, and notably improves with the use of linear binning.

We next examine real-world measurements from the Palmer Penguins dataset [10]. We estimate densities for penguin body mass (in grams) for $n = 123$ Gentoo penguins on the domain $[0, 7000]$. Figure 3 shows maximum estimation errors. We again see that the combination of Deriche approximation and linear binning produces the best results, often with sub-pixel error. Figure 4 shows a subset of the visualized density. The box filter methods again produce perceptible deviations, which in multiple instances obscure local extrema. Deriche approximation produces a pixel-perfect result.

To assess scalability, we generate datasets of arbitrary size based on the Gentoo penguins data. We first fit a kernel density estimate using direct calculation ($\sigma = 200$, based on the NRD value of 204.11), then sample from the resulting distribution to generate datasets ranging from $n = 100$ to $n = 10$M points. Each timing measurement is taken for $m = 512$ bins and averages runs for five bandwidths (100, 150, 200, 250, 300) centered near the original NRD value. Figure 5 plots the results. For small n, box filtering is slightly faster as it involves fewer arithmetic operations. As n increases, the $O(n)$ binning calculation dominates and all methods exhibit similar performance.

4.3 Results: 2D Estimation

To assess bivariate estimates, we use the classic cars dataset [16] and examine the relationship between mileage and horsepower. We use the same error measure. Figure 6 presents the error across binning and bandwidth choices (the same bandwidth is used for the x- and y-dimensions), with similar patterns as before. Deriche approximation with linear binning at $m = 512$ bins produces notably low error rates.

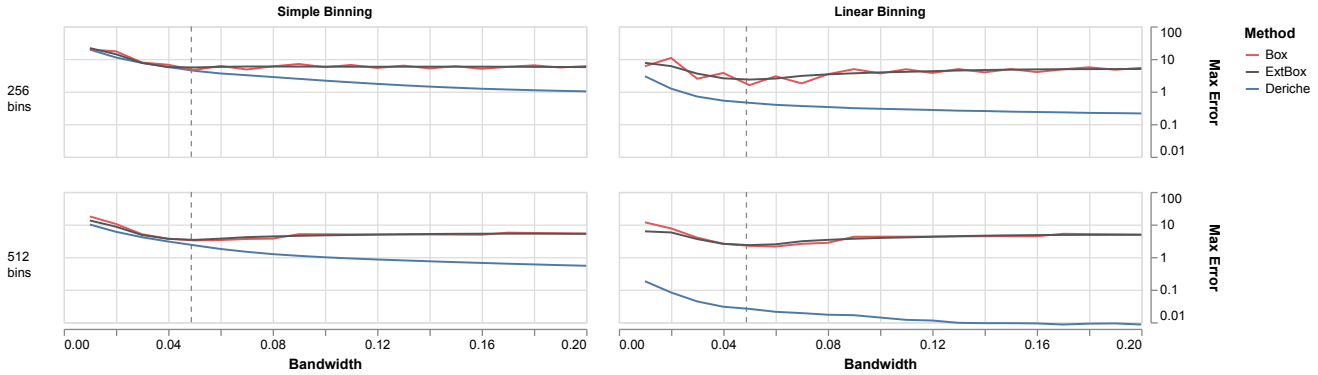

Figure 6: 2D estimation error for car data. Error (on a log scale) is measured as the maximum pixel error given a 100-pixel plot height. Dashed gray lines indicate the NRD σ value. With 512 bins and linear binning, the Deriche method results in sub-pixel accuracy at all sampled bandwidths.

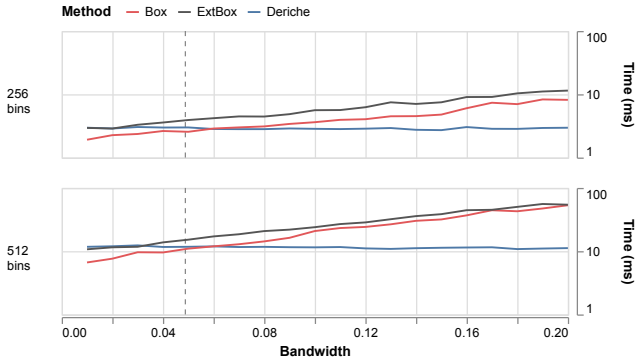

Figure 7: Running time of 2D estimation on car data, by bandwidth. At low bandwidths, Deriche's method is slightly slower due to more arithmetic operations. As the bandwidth increases, the box filters require larger grids, leading to longer running times.

Figure 8 shows heatmaps and contour plots for 2D density estimation, both as separate plots and as contours overlaid on a ground truth heatmap. We set $\sigma = 0.04$ for both the x- and y-dimensions, near the NRD estimates (0.049, 0.048) for each variable. The same contour thresholds – $[0.01, 0.08]$ with increments of 0.01 – are applied for each method. Comparison of the contour plots reveals hallucinators [12], where approximation methods produce different visual features for the same underlying data. The Deriche method provides a pixel-perfect match to the true density, but the box filter methods result in different extrema as well as distorted contour shapes.

As shown earlier (Figure 5), for large datasets the running time of binned KDE methods is dominated by the $O(n)$ binning. Here we instead assess the effect of bandwidth on 2D estimation time, shown in Figure 7. At low bandwidths, standard box filtering is fastest due to fewer operations per bin. However, both box filter methods become slower at larger bandwidths due to the need to enlarge the underlying grid. This overhead is exacerbated for 2D estimation, as the number of expanded cells multiply across grid rows and columns. In contrast the Deriche method is stable across bandwidths as it does not require grid extensions, with performance matching or exceeding the other methods for bandwidths at or above the NRD bandwidth suggestion.

5 Conclusion

We survey approaches for KDE, finding that a combination of linear binning and Deriche approximation results in fast, linear-time performance and excellent accuracy at all but the smallest bandwidths. Though limited to Gaussian kernels only, this approach provides fast and accurate KDE for the tested univariate and bivariate visualizations. Our implementation and benchmarks, including additional error metrics, are available as open source software

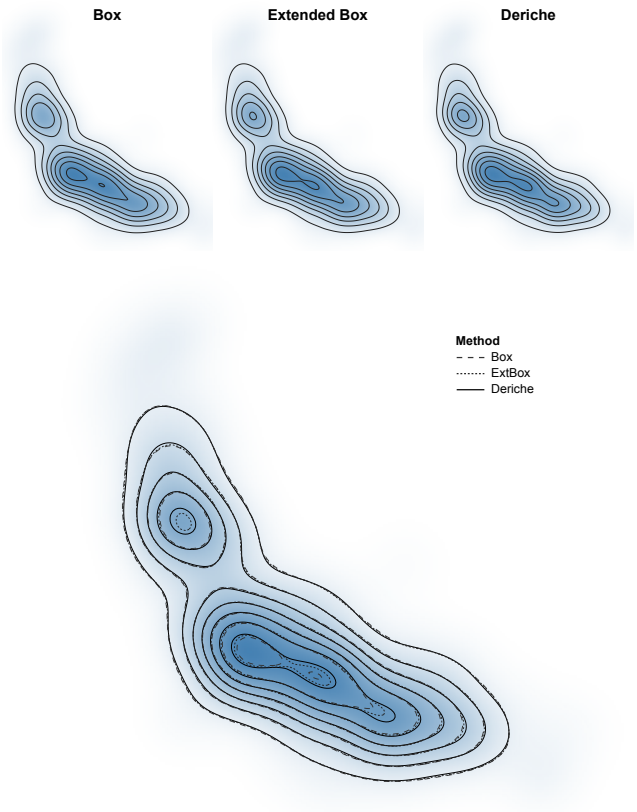

Figure 8: Heatmaps and contour plots of car data (miles per gallon vs. horsepower). *Top:* plots per density method. *Bottom:* contour lines per method overlaid for comparison. Deriche's approximation matches the precise density estimate. Box filters result in extra or missing contour lines and distorted shapes.

(`https://github.com/uwdata/fast-kde`) and we intend to integrate this method into existing web-based visualization libraries.

Acknowledgments

We thank the UW Interactive Data Lab and anonymous reviewers. The work was supported by a Moore Foundation software grant.

References

[1] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[2] M. Bullmann, T. Fetzer, F. Ebner, F. Deinzer, and M. Grzegorzek. Fast Kernel Density Estimation Using Gaussian Filter Approximation. In *Proc. International Conference on Information Fusion (FUSION)*, pp. 1233–1240, 2018.

[3] M. Correll and M. Gleicher. Error Bars Considered Harmful: Exploring Alternate Encodings for Mean and Error. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2142–2151, 2014.

[4] R. Deriche. Fast Algorithms for Low-Level Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):78–87, 1990.

[5] R. Deriche. Recursively implementing the Gaussian and its derivatives. techreport, INRIA, 1993.

[6] P. Getreuer. A Survey of Gaussian Convolution Algorithms. *Image Processing On Line*, 3:286–310, 2013.

[7] A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In *Proc. 2003 SIAM International Conference on Data Mining*, pp. 203–211, 2003.

[8] P. Gwosdek, S. Grewenig, A. Bruhn, and J. Weickert. Theoretical foundations of Gaussian convolution by extended box filtering. In *Proc. International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 447–458, 2011.

[9] J. L. Hintze and R. D. Nelson. Violin Plots: A Box Plot-Density Trace Synergism. *The American Statistician*, 52(2):181–184, 1998.

[10] A. M. Horst, A. P. Hill, and K. B. Gorman. palmerpenguins: Palmer Archipelago (Antarctica) penguin data. Technical report, 2020. R package version 0.1.0.

[11] M. C. Jones and H. W. Lotwick. On the errors involved in computing the empirical characteristic function. *Journal of Statistical Computation and Simulation*, 17(2):133–149, 1983.

[12] G. Kindlmann and C. Scheidegger. An Algebraic Process for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2181–2190, 2014.

[13] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3d Surface Construction Algorithm. *SIGGRAPH Computer Graphics*, 21(4):163–169, 8 1987.

[14] E. Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065 – 1076, 1962.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] E. Ramos and D. Donoho. Asa Data Exposition Dataset, 1983.

[17] M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[18] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2016.

[19] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive Vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, 2015.

[20] D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley Sons, 1992.

[21] D. W. Scott and S. J. Sheather. Kernel density estimation with binned data. *Communications in Statistics - Theory and Methods*, 14(6):1353–1359, 1985.

[22] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society, Series B (Methodological)*, 53(3):683–690, 1991.

[23] B. W. Silverman. Algorithm AS 176: Kernel density estimation using the fast Fourier transform. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 31(1):93–99, 1982.

[24] W. N. Venables and B. D. Ripley. *Modern applied statistics with S*. Springer, 2002.

[25] M. P. Wand. Fast Computation of Multivariate Kernel Estimators. *Journal of Statistical Computation and Simulation*, 3(4):433–445, 1994.

[26] W. M. Wells. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):234–239, 1986.

[27] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.