Fast Kernel Density Estimation using Gaussian Filter Approximation

Markus Bullmann, Toni Fetzer, Frank Ebner, and Frank Deinzer
Faculty of Computer Science and Business Information Systems
University of Applied Sciences Würzburg-Schweinfurt
Würzburg, Germany
{markus.bullmann, toni.fetzer, frank.ebner, frank.deinzer}@fhws.de

Marcin Grzegorzek
Pattern Recognition Group
University of Siegen
Siegen, Germany
marcin.grzegorzek@uni-siegen.de

***Abstract*—It is common practice to use a sample-based representation to solve problems having a probabilistic interpretation. In many real world scenarios one is then interested in finding a best estimate of the underlying problem, e. g. the position of a robot. This is often done by means of simple parametric point estimators, providing the sample statistics. However, in complex scenarios this frequently results in a poor representation, due to multimodal densities and limited sample sizes.**

Recovering the probability density function using a kernel density estimation yields a promising approach to solve the state estimation problem i. e. finding the "real" most probable state, but comes with high computational costs. Especially in time critical and time sequential scenarios, this turns out to be impractical. Therefore, this work uses techniques from digital signal processing in the context of estimation theory, to allow rapid computations of kernel density estimates. The gains in computational efficiency are realized by substituting the Gaussian filter with an approximate filter based on the box filter. Our approach outperforms other state of the art solutions, due to a fully linear complexity and a negligible overhead, even for small sample sets. Finally, our findings are evaluated and tested within a real world sensor fusion system.

I. Introduction

Sensor fusion approaches are often based upon probabilistic descriptions like particle filters, using samples to represent the distribution of a dynamical system. To update the system recursively in time, probabilistic sensor models process the noisy measurements and a state transition function provides the system's dynamics. Therefore a sample or particle is a representation of one possible system state, e. g. the position of a pedestrian within a building. The set of all particles represents the posterior of the system. In most real world scenarios one is then interested in finding the most probable state within the state space, to provide the best estimate of the underlying problem, generally speaking, solving the state estimation problem. In the discrete manner of a sample representation this is often done by providing a single value, also known as sample statistic, to serve as a "best guess".

This value is often calculated by means of simple parametric point estimators, e. g. the weighted-average over all samples or, the sample with the highest weight [1]. While such point estimators are computational fast and suitable most of the time, it is not uncommon that they fail to recover the state in more complex scenarios. Especially time-sequential, non-linear and non-Gaussian state spaces, depending upon a high number of different sensor types, frequently suffer from a multimodal representation of the posterior distribution. For example, in a localization scenario where a bimodal distribution represents the current posterior, a reliable position estimation is more likely to be at one of the modes, instead of somewhere in-between, like provided by a simple weighted-average estimation. Additionally, in most practical scenarios the sample size, and hence the resolution is limited, causing the variance of the sample based estimate to be high [2].

It is obvious, that a computation of the full posterior could solve the above, but finding such an analytical solution is an intractable problem, which is the reason for applying a sample representation in the first place. A feasible alternative is to estimate the parameters of a specific parametric model based on the sample set, assuming that the unknown distribution is approximately a parametric distribution, e. g. like the normal distribution. Given the estimated parameters the most probable state can be obtained from the parameterized density function. However, parametric models fail when the assumption does not fit the underlying model. For many real world problems assuming a parametric distribution is too limiting as the posterior is changing over time. As a result, those techniques are not able to provide an accurate statement about the most probable state, rather causing misleading or false outcomes.

Another promising way is to recover the probability density function from the sample set itself, by using a non-parametric estimator like a kernel density estimation (KDE). With this, the "real" most probable state is given by the maximum of the density estimation and thus avoids the aforementioned drawbacks. However, non-parametric estimators tend to consume a large amount of computation time, which renders them unpractical for real time scenarios. Nevertheless, the availability of a fast processing non-parametric density estimate might improve the accuracy of today's sensor fusion systems without sacrificing their real time capability.

In this paper, a novel approximation approach for rapid computation of the KDE is presented. The basic idea is to interpret the density estimation problem as a filtering operation. We show that computing the KDE with a Gaussian kernel on binned data is equal to applying a Gaussian filter on the binned data. This allows us to use a well known approximation

scheme based on multiple recursions of a box filter, which yields an approximative Gaussian filter given by the central limit theorem [3].

This process converges quite fast to a reasonable close approximation of the ideal Gaussian. In addition, a box filter can be computed extremely fast by a computer, due to its intrinsic simplicity. While the idea to use several box filter passes to approximate a Gaussian has been around for a long time, the application to obtain a fast KDE is new. Especially in time critical and time sequential sensor fusion scenarios, the here presented approach outperforms other state of the art solutions, due to a fully linear complexity and a negligible overhead, even for small sample sets. In addition, it requires only a few elementary operations and is highly parallelizable.

II. Related work

The kernel density estimator is a well known non-parametric density estimator, originally described independently by Rosenblatt [4] and Parzen [5]. It was subject to extensive research and its theoretical properties are well understood. A comprehensive reference is given by Scott [6]. Although classified as non-parametric, the KDE depends on two free parameters, the kernel function and its bandwidth. The selection of a "good" bandwidth is still an open problem and heavily researched. An extensive overview regarding the topic of automatic bandwidth selection is given by [7].

The great flexibility of the KDE makes it suitable for many applications. However, this comes at the cost of a slow computation speed. The complexity of a naive implementation of the KDE is $\mathcal{O}(MN)$, given by M evaluations of N data samples as input size. Various methods have been proposed to reduce the computation time of the KDE.

An obvious way to speed up the computation is to reduce the number of evaluated kernel functions. One possible optimization is based on k-nearest neighbour search, performed on spatial data structures. These algorithms reduce the number of evaluated kernels by taking the distance between clusters of data points into account [8].

Another approach is to reduce the algorithmic complexity of the sum over Gaussian functions, by employing a specialized variant of the fast multipole method. The term fast Gauss transform was coined by Greengard [9] who suggested this approach to reduce the complexity to $\mathcal{O}(N+M)$.

Recent methods based on the self-consistent KDE proposed by Bernacchia and Pigolotti [10] allow to obtain an estimate without any assumptions, i. e. the kernel and bandwidth are both derived during the estimation. They define a Fourier-based filter on the empirical characteristic function of a given dataset. The computation time was further reduced by O'Brien et al. using a non-uniform fast Fourier transform (FFT) algorithm to efficiently transform the data into Fourier space [11].

In general, it is desirable to compute the estimate directly from the sample set. However, reducing the sample size by distributing the data on an equidistant grid can significantly reduce the computation time, if an approximative KDE is acceptable. Silverman [12] originally suggested to combine adjacent data points into data bins, which results in a discrete convolution structure of the KDE. Allowing to efficiently compute the estimate using a FFT algorithm. This approximation scheme was later called binned KDE (BKDE) and was extensively studied [13] [14] [15]. While the FFT algorithm constitutes an efficient algorithm for large sample sets, it adds an noticeable overhead for smaller ones.

The idea to approximate a Gaussian filter using several box filters was first formulated by Wells [16]. Kovesi [3] suggested to use two box filters with different widths to increase accuracy maintaining the same complexity. To eliminate the approximation error completely, Gwosdek et al. [17] proposed a new approach called extended box filter.

This work highlights the discrete convolution structure of the BKDE and elaborates its connection to digital signal processing, especially the Gaussian filter. Accordingly, this results in an equivalence relation between BKDE and Gaussian filter. It follows, that the above mentioned box filter approach is also an approximation of the BKDE, resulting in an efficient computation scheme presented within this paper. This approach has a lower complexity as comparable FFT-based algorithms and adds only a negligible small error, while improving the performance significantly.

III. Kernel Density Estimator

The KDE is often the preferred tool to estimate a density function from discrete data samples because of its flexibility and ability to produce a continuous estimate. Given an univariate random sample set $X = \{X_1, \dots, X_N\}$, where X has the density function f and let $w_1, \dots w_N$ be associated weights. The kernel estimator $\hat{f}$ which estimates f at the point x is given as

$$\hat{f}(x) = \frac{1}{W} \sum_{i=1}^{N} \frac{w_i}{h} K\left(\frac{x - X_i}{h}\right), \tag{1}$$

where $W = \sum_{i=1}^{N} w_i$ and $h \in \mathbf{R}^+$ is an arbitrary smoothing parameter called bandwidth. K is a kernel function such that $\int K(u)\,\mathrm{d}u = 1$. In general, any kernel can be used, however a common advice is to chose a symmetric and low-order polynomial kernel. Several popular kernel functions are used in practice, like the Uniform, Gaussian, Epanechnikov, or Silverman kernel [6].

While the kernel estimate inherits all the properties of the kernel, usually it is not of crucial matter if a non-optimal kernel was chosen. As a matter of fact, the quality of the kernel estimate is primarily determined by the smoothing parameter h [6]. It is common practice to suspect that the data is approximately Gaussian, hence the Gaussian kernel is frequently used. In this work we choose the Gaussian kernel in favour of computational efficiency as our approach is based on the approximation of the Gaussian filter. The Gaussian kernel is given as

$$K_G(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right). \tag{2}$$

The flexibility of the KDE comes at the expense of computation speed, which leads to the development of more efficient computation schemes. The computation time depends, besides the number of calculated points M, on the input size, namely the size of sample N. In general, reducing the size of the sample set negatively affects the accuracy of the estimate. Still, N is a suitable parameter to speed up the computation.

The BKDE reduces N by combining each single sample with its adjacent samples into bins, and thus, approximates the KDE. Each bin represents the count of the sample set at a given point of an equidistant grid with spacing δ. A binning rule distributes each sample among the grid points $g_j = j\delta$, indexed by $j \in \mathbf{Z}$. Computation requires a finite grid on the interval $[a, b]$ containing the data, thus the number of grid points is $G = (b - a)/\delta + 1$ [15].

Given a binning rule r_j the BKDE $\tilde{f}$ of a density f computed pointwise at the grid point g_x is given as

$$\tilde{f}(g_x) = \frac{1}{W} \sum_{j=1}^{G} \frac{C_j}{h} K\left(\frac{g_x - g_j}{h}\right), \tag{3}$$

where G is the number of grid points and

$$C_j = \sum_{i=1}^{N} r_j(x_i, \delta) \tag{4}$$

is the count at grid point g_j, such that $\sum_{j=1}^{G} C_j = W$ [15].

In theory, any function which determines the count at grid points is a valid binning rule. However, for many applications it is recommend to use the simple binning rule

$$r_j(x, \delta) = \begin{cases} w_j & \text{if } x \in ((j - \frac{1}{2})\delta, (j + \frac{1}{2})\delta] \\ 0 & \text{else} \end{cases} \tag{5}$$

or the common linear binning rule, which divides the sample into two fractional weights shared by the nearest grid points

$$r_j(x, \delta) = \begin{cases} w_j(1 - |\delta^{-1}x - j|) & \text{if } |\delta^{-1}x - j| \leq 1 \\ 0 & \text{else.} \end{cases} \tag{6}$$

An advantage is that their impact on the approximation error is extensively investigated and well understood [15]. Both methods can be computed with a fast $\mathcal{O}(N)$ algorithm, as simple binning is essentially the quotient of an integer division and the fractional weights of the linear binning are given by the remainder of the division. As linear binning is more precise, it is often preferred over simple binning [13].

While linear binning improves the accuracy of the estimate, the choice of the grid size is of more importance. The number of grid points G determines the trade-off between the approximation error caused by the binning and the computational speed of the algorithm. Clearly, a large value of G produces an estimate close to the regular KDE, but requires more evaluations of the kernel compared to a coarser grid. However, it is unknown what particular G gives the best trade-off for any given sample set. In general, there is no definite answer because the amount of binning depends on the structure of the unknown density and the sample size [15].

A naive implementation of (3) reduces the number of kernel evaluations to $\mathcal{O}(G^2)$, assuming that $G < N$ [13]. However, due to the fixed grid spacing several kernel evaluations are the same and can be reused. This reduces the number of kernel evaluations to $\mathcal{O}(G)$, but the number of additions and multiplications required are still $\mathcal{O}(G^2)$. Using the FFT to perform the discrete convolution, the complexity can be further reduced to $\mathcal{O}(G \log G)$ [12].

The FFT-convolution approach is usually highlighted as the striking computational benefit of the BKDE. However, for this work it is key to recognize the discrete convolution structure of (3), as this allows to interpret the computation of a density estimate as a signal filter problem. This makes it possible to apply a wide range of well studied techniques from the broad field of digital signal processing (DSP). Using the Gaussian kernel from (2) in conjunction with (3) gives

$$\tilde{f}(g_x) = \frac{1}{W\sqrt{2\pi}} \sum_{j=1}^{G} \frac{C_j}{h} \exp\left(-\frac{(g_x - g_j)^2}{2h^2}\right). \tag{7}$$

The above formula is a convolution of the data and the Gaussian kernel. More precisely, it is a discrete convolution of the finite data grid and the Gaussian function. In terms of DSP this is analogous to filter the binned data with a Gaussian filter. This finding allows to speed up the computation of the density estimate by using a fast approximation scheme based on iterated box filters.

IV. Gaussian Filter Approximation

Digital filters are implemented by convolving the input signal with a filter kernel, i. e. the digital filter's impulse response. Consequently, the filter kernel of a Gaussian filter is a Gaussian with finite support [18]. Assuming a finite-support Gaussian filter kernel of size M and an input signal x, discrete convolution produces the smoothed output signal

$$(G_\sigma * x)(i) = \frac{1}{\sigma\sqrt{2\pi}} \sum_{k=0}^{M-1} x(k) \exp\left(-\frac{(i - k)^2}{2\sigma^2}\right), \tag{8}$$

where σ is a smoothing parameter called standard deviation.

Note that (7) has the same structure as (8), except the varying notational symbol of the smoothing parameter and the different factor in front of the sum. While in both equations the constant factor of the Gaussian is removed from the inner sum, (7) has an additional normalization factor W^{-1}. This factor is necessary to ensure that the estimate is a valid density function, i. e. that it integrates to one. Such a restriction is superfluous in the context of digital filters, so the normalization factor is omitted.

Computation of a digital filter using the naive implementation of the discrete convolution algorithm yields $\mathcal{O}(NM)$, where N is again the input size given by the length of the input signal and M is the size of the filter kernel. In order to capture all significant values of the Gaussian function the kernel size M must be adopted to the standard deviation of the Gaussian. A faster $\mathcal{O}(N)$ algorithm is given by the well-known approximation scheme based on iterated box filters.

Algorithm 1: Recursive 1D box filter

Input: f defined on $[1, N]$ and box radius l
Output: $u \leftarrow B_L * f$
1: $a := \sum_{i=-l}^{l} f(i)$
2: $u(1) := a/(2l+1)$
3: **for** $i = 2$ **to** N **do**
4: $\quad a := a + f(i+l) - f(i-l-1)$ ▷ (11)
5: $\quad u(i) := a/(2l+1)$
6: **end for**

While reducing the algorithmic complexity this approximation also reduces computational time significantly due to the simplistic computation scheme of the box filter. Following that, an implementation of the iterated box filter is one of the fastest ways to obtain an approximative Gaussian filtered signal.

The box filter is a simplistic filter defined as convolution of the input signal and the box (or rectangular) function. A discrete box filter of radius $l \in \mathbf{N}_0$ is given as

$$(B_L * x)(i) = \sum_{k=0}^{L-1} x(k) B_L(i-k) \tag{9}$$

where $L = 2l + 1$ is the width of the filter and

$$B_L(i) = \begin{cases} L^{-1} & \text{if } -l \leq i \leq l \\ 0 & \text{else} \end{cases} \tag{10}$$

is the discrete box kernel.

Such a filter clearly requires $\mathcal{O}(NL)$ operations, where N is the input signal length. It is well-known that a box filter can approximate a Gaussian filter by repetitive recursive computations. Given by the central limit theorem of probability, repetitive convolution of a rectangular function with itself eventually yields a Gaussian in the limit. Likewise, filtering a signal with the box filter several times approximately converges to a Gaussian filter. In practice three to five iterations are most likely enough to obtain a reasonable close Gaussian approximation [3].

This allows rapid approximation of the Gaussian filter using iterated box filter, which only requires a few addition and multiplication operations. Opposed to the Gaussian filter, where several evaluations of the exponential function are necessary. The most efficient way to compute a single output value of the box filter is:

$$y(i) = y(i-1) + x(i+l) - x(i-l-1) \tag{11}$$

This recursive calculation scheme further reduces the time complexity of the box filter to $\mathcal{O}(N)$, by reusing already calculated values. Furthermore, only one addition and subtraction is required to calculate a single output value. The overall algorithm to efficiently compute (9) is listed in Algorithm 1.

Given a fast approximation scheme, it is necessary to construct a box filter analogous to a given Gaussian filter. As seen in (8), the sole parameter of the Gaussian kernel is the standard deviation σ. In contrast, the box function (10) is parametrized by its width L. Therefore, in order to approximate the Gaussian filter of a given σ, a corresponding value of L must be found. Given n iterations of box filters with identical sizes the ideal size L_{ideal}, as suggested by Wells [16], is

$$L_{\text{ideal}} = \sqrt{\frac{12\sigma^2}{n} + 1} \quad . \tag{12}$$

In general L_{ideal} can be any real number, but B_L in (10) is restricted to odd integer values. Hence, the set of possible approximated standard deviations is limited, because the ideal width must be rounded to the next valid value. In order to reduce the rounding error Kovesi [3] proposes to perform two box filters with different widths

$$\begin{aligned} L_1 &= \begin{cases} \lfloor L_{\text{ideal}} \rfloor - 1 & \text{if } \lfloor L_{\text{ideal}} \rfloor \text{ is odd} \\ \lfloor L_{\text{ideal}} \rfloor & \text{else} \end{cases} \\ L_2 &= L_1 + 2 \quad . \end{aligned} \tag{13}$$

Given L_1 and L_2 the approximation is done by computing m box filters of width L_1 followed by $(n-m)$ box filters of size L_2, where $0 \leq m \leq n$. As all other values are known, m can be computed with

$$m = \frac{12\sigma^2 - nL_1^2 - 4nL_1 - 3n}{-4L_1 - 4} \quad . \tag{14}$$

The approximated σ as a function of the integer width has a staircase shaped graph. By reducing the rounding error, the step size of the function is reduced. However, the overall shape will not change. Gwosdek et al. [17] proposed an approach which allows to approximate any real-valued value of σ. Just like the conventional box filter, the extended version has a uniform value in the range $[-l; l]$, but unlike the conventional, the extended box filter has different values at its edges. This extension introduces only marginal computational overhead over conventional box filtering.

V. Extension to multi-dimensional data

So far only univariate sample sets were considered. This is due to the fact, that the equations of the KDE (1), BKDE (3), Gaussian filter (8), and the box filter (9) are quite easily extended to multi-dimensional input. Each method can be seen as several one-dimensional problems combined to a multi-dimensional result. In the following, the generalization to multi-dimensional input are briefly outlined.

In order to estimate a multivariate density using KDE or BKDE, a multivariate kernel needs to be used. Multivariate kernel functions can be constructed in various ways, however, a popular way is given by the product kernel. Such a kernel is constructed by combining several univariate kernels into a product, where each kernel is applied in each dimension with a possibly different bandwidth.

Given a multivariate random variable $\boldsymbol{X} = (x_1, \ldots, x_d)^T$ in d dimensions. The sample set $\mathcal{X} = (x_{i,j}) = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$

is a $n \times d$ matrix [6]. The multivariate KDE $\hat{f}$ which defines the estimate pointwise at $\boldsymbol{u} = (u_1, \ldots, u_d)^T$ is given as

$$\hat{f}(\boldsymbol{u}) = \frac{1}{W} \sum_{i=1}^{n} \frac{w_i}{h_1 \ldots h_d} \left[\prod_{j=1}^{d} K\left(\frac{u_j - x_{i,j}}{h_j} \right) \right], \qquad (15)$$

where the bandwidth is given as a vector $\boldsymbol{h} = (h_1, \ldots, h_d)$.

Note that (15) does not include all possible multivariate kernels, such as spherically symmetric kernels, which are based on rotation of an univariate kernel. In general, a multivariate product and spherically symmetric kernel based on the same univariate kernel will differ. The only exception is the Gaussian kernel, which is spherically symmetric and has independent marginals. In addition, only smoothing in the direction of the axes is possible. If smoothing in other directions is necessary, the computation needs to be done on a prerotated sample set and the estimate needs to be rotated back to fit the original coordinate system [14].

For the multivariate BKDE, in addition to the kernel function, the grid and the binning rules need to be extended to multivariate data, which is rather straightforward, as the grid is easily defined on many dimensions. Likewise, the common and linear binning rule scale with dimensionality [14].

In general, multi-dimensional filters are multi-dimensional convolution operations. However, by utilizing the separability property of convolution, a straightforward and a more efficient implementation can be found. Convolution is separable if the filter kernel is separable, i. e. it can be split into successive convolutions of several kernels. In example, the Gaussian filter is separable, because of $e^{x^2+y^2} = e^{x^2} \cdot e^{y^2}$. Likewise digital filters based on such kernels are called separable filters. They are easily applied to multi-dimensional signals, because the input signal can be filtered in each dimension individually by an one-dimensional filter [18].

VI. Usage

Consider a set of two-dimensional samples with associated weights, e. g. generated from a particle filter system. The overall process for bivariate data is described in Algorithm 2.

Assuming that the given N samples are stored in a sequential list, the first step is to create a grid representation. In order to efficiently construct the grid and to allocate the required memory, the extrema of the samples in each dimension need to be known in advance. These limits might be given by the application. For example, the position of a pedestrian within a building is limited by the physical dimensions of the building. Such knowledge should be integrated into the system to avoid a linear search over the sample set, naturally reducing the computation time.

Given the extreme values of the samples and grid sizes G_1 and G_2 defined by the user, a $G_1 \times G_2$ grid can be constructed, using a binning rule from (5) or (6). As the number of grid points directly affects both, computation time and accuracy, a suitable grid should be as coarse as possible, but at the same time narrow enough to produce an estimate sufficiently fast with an acceptable approximation error.

Algorithm 2: Bivariate BOXKDE

Input: Samples $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ and weights $w_1, \ldots, w_N$
Output: Approximative density estimate $\hat{f}$ on $G_1 \times G_2$

1: **for** $i = 1$ **to** N **do** ▷ Data binning
2: Find the 4 nearest grid points to $\boldsymbol{X}_i$
3: Compute bin count $C_{i,j}$ as recommended by [14]
4: **end for**

5: $\tilde{\boldsymbol{h}} := \boldsymbol{\delta}^{-1}\boldsymbol{h}$ ▷ Scaled bandwidth
6: $\boldsymbol{L} := \lfloor \sqrt{12\tilde{\boldsymbol{h}}^2 n^{-1} + \mathbf{1}} \rfloor$ ▷ (12)

7: **loop** n **times** ▷ n separated box filter iterations
8: **for** $i = 1$ **to** G_1 **do**
9: Compute $\hat{f}_{i,1:G_2} \leftarrow B_{L_2} * C_{i,1:G_2}$ ▷ Alg. 1
10: **end for**
11: **for** $j = 1$ **to** G_2 **do**
12: Compute $\hat{f}_{1:G_1,j} \leftarrow B_{L_1} * C_{1:G_1,j}$ ▷ Alg. 1
13: **end for**
14: **end loop**

If the extreme values are known in advanced, the computation of the grid is $\mathcal{O}(N)$, otherwise an additional $\mathcal{O}(N)$ search is required. The grid is stored as a linear array in memory, thus its space complexity is $\mathcal{O}(G_1 \cdot G_2)$.

Next, the binned data is filtered with a Gaussian using the box filter approximation. The box filter's width is derived by (12) from the standard deviation of the approximated Gaussian, which is in turn equal to the bandwidth of the KDE. However, the bandwidth h needs to be scaled according to the grid size. This is necessary as h is defined in the input space of the KDE, i. e. in relation to the sample data. In contrast, the bandwidth of a BKDE is defined in the context of the binned data, which differs from the unbinned data due to the discretisation of the samples. For this reason, h needs to be divided by the bin size to account the discrepancy between the different sampling spaces.

Given the scaled bandwidth the required box filter's width can be computed. Due to its best runtime performance the recursive box filter implementation is used. If multivariate data is processed, the algorithm is easily extended due to its separability. Each filter pass is computed in $\mathcal{O}(G)$ operations, however, an additional memory buffer is required [18].

While the integer-sized box filter requires fewest operations, it causes a larger approximation error due to rounding errors. Depending on the required accuracy, the extended box filter algorithm can further improve the estimation results with only a small additional overhead [17]. Due to its simple indexing scheme, the recursive box filter can easily be computed in parallel using SIMD operations and parallel computation cores.

Finally, the most likely state can be obtained from the filtered data, i. e. from the estimated discrete density, by searching filtered data for its maximum value. This last step

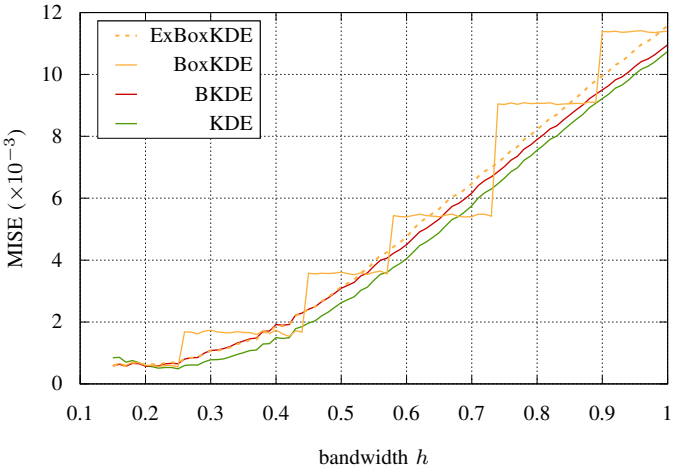

Fig. 1: MISE relative to the ground truth as a function of h. While the error curves of the BKDE (red) and the BoxKDE based on the extended box filter (orange dotted line) resemble the overall course of the error of the KDE (green), the regular BoxKDE (orange) exhibits noticeable jumps due to rounding.

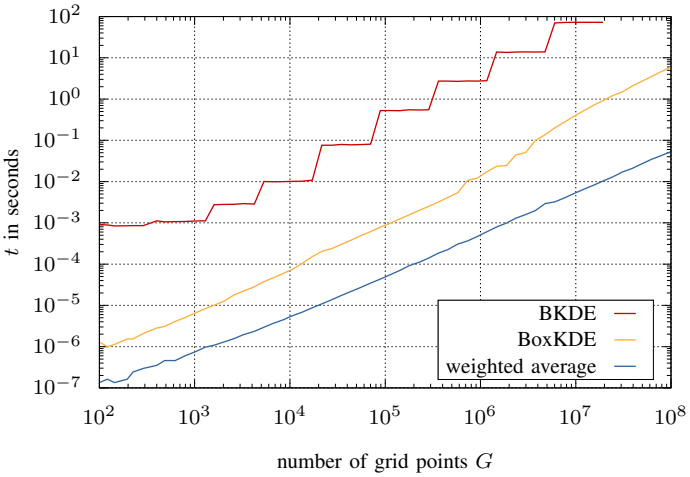

Fig. 2: Logarithmic plot of the runtime performance with increasing grid size G and bivariate data. The weighted-average estimate (blue) performs fastest followed by the BoxKDE (orange) approximation, which is magnitudes slower, especially for $G < 10^3$.

can be integrated into the last filter operation, by recording the largest output value.

VII. Experiments

A. Mean Integrated Squared Error

We empirically evaluate the feasibility of our BoxKDE method by analyzing its approximation error. In order to evaluate the deviation of the estimate from the original density, the mean integrated squared error (MISE) is used. Those errors are compared for the KDE and its various approximations. To match the requirements of our application, a synthetic sample set $\mathcal{X}$ with $N = 5000$ drawn from a bivariate mixture normal density f given by (16) provides the basis of the comparison. For each method the estimate is computed and the MISE relative to f is calculated. The specific structure of the underlying distribution clearly affects the error in the estimate, but only the closeness of our approximation to the KDE is of interest. Hence, f is of minor importance here and was chosen rather arbitrary to highlight the behavior of the BoxKDE.

$$\begin{aligned}\boldsymbol{X} \sim \mathcal{N}\left(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], 0.5\boldsymbol{I}\right) + \mathcal{N}\left(\left[\begin{smallmatrix}3\\0\end{smallmatrix}\right], \boldsymbol{I}\right) + \mathcal{N}\left(\left[\begin{smallmatrix}0\\3\end{smallmatrix}\right], \boldsymbol{I}\right) \\ + \mathcal{N}\left(\left[\begin{smallmatrix}-3\\0\end{smallmatrix}\right], \boldsymbol{I}\right) + \mathcal{N}\left(\left[\begin{smallmatrix}0\\-3\end{smallmatrix}\right], \boldsymbol{I}\right)\end{aligned} \tag{16}$$

Four estimates are computed with varying bandwidth using the KDE, BKDE, BoxKDE, and ExBoxKDE, which uses the extended box filter. All estimates are calculated at 30×30 equally spaced points. The graphs of the MISE between f and the estimates as a function of $h \in [0.15, 1.0]$ are given in fig. 1. A minimum error is obtained with $h = 0.25$, for larger values oversmoothing occurs and the modes gradually fuse together.

Both the BKDE and the ExBoxKDE resemble the error curve of the KDE quite well and stable. They are rather close to each other, with a tendency to diverge for larger h. In contrast, the error curve of the BoxKDE has noticeable jumps at $h = \{0.25, 0.44, 0.57, 0.73, 0.89\}$. These jumps are caused by the rounding of the integer-valued box width given by (12).

As the extended box filter is able to approximate an exact σ, such discontinuities do not appear. Consequently, it reduces the overall error of the approximation, even though only marginal in this scenario. The global average MISE over all values of h is 0.0051 for the regular box filter and 0.0049 in case of the extended version. The global maximum MISE is 0.0011 for both versions. The choice between the extended and regular box filter algorithm depends on how large the acceptable error should be, thus on the particular application.

Other test cases of theoretical relevance are the MISE as a function of the grid size G and the sample size N. However, both cases do not give a deeper insight of the error behavior of our method, as it closely mimics the error curve of the KDE and only confirms theoretical expectations.

B. Performance

In the following, we underpin the promising theoretical linear time complexity of our method with empirical time measurements compared to other methods. All tests are performed on an Intel Core i5-7600K CPU @ 4.2 GHz, and 16 GB main memory. We compare our C++ implementation of the BoxKDE approximation as shown in algorithm 2 to the R language `ks` package, which provides a FFT-based BKDE implementation based on optimized C functions at its core. With state estimation problems in mind, we additionally provide a C++ implementation of a weighted average estimator. An equivalently sized input sample set was used for the weighted average, as its runtime depends on the sample size and not the grid size.

The results of the performance comparison are presented in fig. 2. The linear complexity of the BoxKDE and the weighted average is clearly visible. Especially for small G up to 10000 grid points the BoxKDE is much faster compared to BKDE. Nevertheless, the simple weighted average approach performs the fastest. However, it is obvious that this comes with major disadvantages, like being prone to multimodalities, as discussed in section I.

Looking at fig. 2, the runtime performance of the BKDE approach is increasing in a stepwise manner with growing G. This behavior is caused by the underlying FFT algorithm. The FFT approach requires the input to be always rounded up to a power of two, what then causes a constant runtime behaviour within those boundaries and a strong performance deterioration at corresponding manifolds. The termination of BKDE graph at $G \approx 1.9 \cdot 10^7$ is caused by an out of memory error in the `ks` package for bigger G.

Both discussed Gaussian filter approximations, namely box filter and extended box filter, yield a similar runtime behavior and therefore a similar curve progression. While the average runtime over all values of G for the standard box filter is $0.4092\,\mathrm{s}$, the extended one has an average of $0.4169\,\mathrm{s}$. To disambiguate fig. 2, we only illustrated the results of the BoxKDE with the regular box filter.

The weighted average has the great advantage of being independent of the dimensionality of the input and can be implemented effortlessly. In contrast, the computation of the BoxKDE approach increases exponentially with increasing number of dimensions. However, due to the linear time complexity and the very simple computation scheme, the overall computation time is still sufficiently fast for many applications and much smaller compared to other methods. The BoxKDE approach presents a reasonable alternative to the weighted-average and is easily integrated into existing systems.

In addition, modern CPUs do benefit from the recursive computation scheme of the box filter, as the data exhibits a high degree of spatial locality in memory and the accesses are reliably predictable.

C. Real World

To demonstrate the capabilities of the proposed method a real world scenario was chosen, namely indoor localization. The given problem is to localize a pedestrian walking inside a building. Ebner et al. proposed a method, which incorporates multiple sensors, e. g. Wi-Fi, barometer, step-detection and turn-detection [19]. At a given time t the system estimates a state providing the most probable position of the pedestrian. It is implemented using a particle filter with sample importance resampling and 5000 particles. The dynamics are modelled realistically, which constrains the movement according to walls, doors and stairs.

We arranged a $223\,\mathrm{m}$ long walk within the first floor of a $76 \times 71\,\mathrm{m}$ sized museum, which was built in the 13th century and offers non-optimal conditions for localization. Since this work only focuses on processing a given sample set, further details of the localization system and the described scenario can be looked up in [20] and [21]. The spacing δ of the grid was set to $27\,\mathrm{cm}$ for x and y-direction, resulting in a grid size of $G = 74019$. The bivariate state estimation was calculated whenever a step was recognized, about every $500\,\mathrm{ms}$.

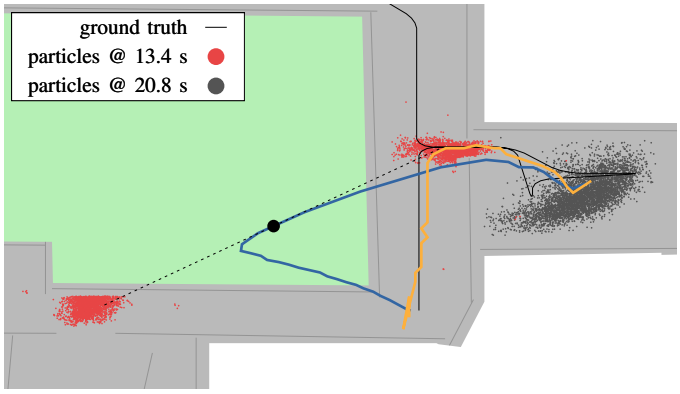

Fig. 3: Occurring bimodal distribution caused by uncertain measurements in the first $13.4\,\mathrm{s}$ of the walk. After $20.8\,\mathrm{s}$, the distribution gets unimodal. The weigted-average estimation (blue) provides a high error compared to the ground truth (solid black), while the BoxKDE approach (orange) does not.

Fig. 3 illustrates a frequently occurring situation, where the particle set splits apart, due to uncertain measurements and multiple possible walking directions. This results in a bimodal posterior distribution, which reaches its maximum distances between the modes at $13.4\,\mathrm{s}$ (black dotted line). Thus estimating the most probable state over time using the weighted-average results in the blue line, describing the pedestrian's position to be somewhere outside the building (light green area). In contrast, the here proposed method (orange line) is able to retrieve a good estimate compared to the ground truth path shown by the black solid line. Due to a right turn, the distribution gets unimodal after $20.8\,\mathrm{s}$. This happens since the lower red particles are walking against a wall, and therefore are punished with a low weight.

This example highlights the main benefits using our approach. While being fast enough to be computed in real time, the proposed method reduces the estimation error of the state in this situation, as it is possible to distinguish the two modes of the density. It is clearly visible, that this enables the system to recover the real state if multimodalities arise. However, in situations with highly uncertain measurements, the estimation error could further increase since the real estimate is not equal to the best estimate, i. e. the real position of the pedestrian.

The error over time for different estimation methods of the complete walk can be seen in fig. 4. It is given by calculating the distance between estimation and ground truth at a specific time t. Estimates provided by simply choosing the maximum particle stand out the most. As expected beforehand, this method provides many strong peaks through continuously jumping between single particles.

Further investigating fig. 4, the BoxKDE performs slightly better than the weighted-average in this specific Monte Carlo

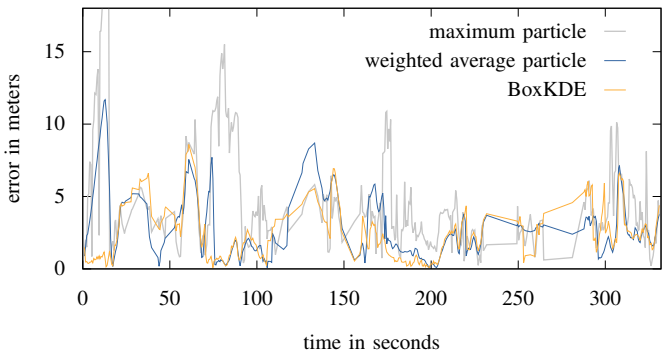

Fig. 4: Error development over time of a single Monte Carlo run of the walk calculated between estimation and ground truth. Between 230 s and 290 s to pedestrian was not moving.

run. However after deploying 100 Monte Carlo runs, the difference becomes insignificant. The main reason for this are again multimodalities caused by faulty or delayed measurements, especially when entering or leaving rooms. Within our experiments the problem occurred due to slow and attenuated Wi-Fi signals inside thick-walled rooms. While the system's dynamics are moving the particles outside, the faulty Wi-Fi readings are holding back a majority by assigning corresponding weights. Therefore, the average between the modes of the distribution is often closer to the ground truth as the real estimate, which is located on the "wrong" mode. With new measurements coming from the hallway or other parts of the building, the distribution and thus the estimation are able to recover.

Nevertheless, it can be seen that our approach is able to resolve multimodalities even under real world conditions. It does not always provide the lowest error, since it depends more on an accurate sensor model than a weighted-average approach, but it is very suitable as a good indicator about the real performance of a sensor fusion system. In the here shown examples we only searched for a global maximum, even though the BoxKDE approach opens a wide range of other possibilities for finding a best estimate.

VIII. Conclusion

Within this paper a novel approach for rapid approximation of the KDE was presented. This is achieved by considering the discrete convolution structure of the BKDE and thus elaborating its connection to digital signal processing, especially the Gaussian filter. Using a box filter as an appropriate approximation results in an efficient computation scheme with a fully linear complexity and a negligible overhead, as demonstrated by the experiments.

The analysis of the error showed that the method shows an similar error behaviour compared to the BKDE. In terms of calculation time, our approach outperforms other state of the art implementations. Despite being more efficient than other methods, the algorithmic complexity still increases in its exponent with an increasing number of dimensions.

Finally, such a fast approximation scheme makes the KDE more attractive for real time use cases. In a sensor fusion context, the availability of a reconstructed density of the posterior enables many new approaches and techniques for finding a best estimate of the system's current state.

References

[1] T. Fetzer, F. Ebner, L. Köping, M. Grzegorzek, and F. Deinzer, "On Monte Carlo Smoothing in Multi Sensor Indoor Localisation," in *Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2016.

[2] V. Verma, S. Thrun, and R. Simmons, "Variable resolution particle filter," in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2003, pp. 976–984.

[3] P. Kovesi, "Fast almost-gaussian filtering," in *Proceedings of the 2010 Int. Conf. on Digital Image Computing: Techniques and Applications*. IEEE, 2010, pp. 121–125.

[4] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[5] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[6] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*, 2nd ed., ser. Wiley series in probability and statistics. Wiley, 2015.

[7] N.-B. Heidenreich, A. Schindler, and S. Sperlich, "Bandwidth selection for kernel density estimation: A review of fully automatic selectors," *AStA Advances in Statistical Analysis*, vol. 97, no. 4, pp. 403–433, 2013.

[8] A. G. Gray and A. W. Moore, "Nonparametric density estimation: Toward computational tractability," in *Proceedings of the 2003 SIAM Int. Conf. on Data Mining*. SIAM, 2003, pp. 203–211.

[9] L. Greengard and J. Strain, "The fast gauss transform," *SIAM Journal on Scientific and Statistical Computing*, vol. 12, no. 1, pp. 79–94, 1991.

[10] A. Bernacchia and S. Pigolotti, "Self-consistent method for density estimation," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 3, pp. 407–422, 2011.

[11] T. A. O'Brien, K. Kashinath, N. R. Cavanaugh, W. D. Collins, and J. P. O'Brien, "A fast and objective multidimensional kernel density estimation method: fastkde," *Computational Statistics & Data Analysis*, vol. 101, pp. 148–160, 2016.

[12] B. Silverman, "Algorithm as 176: Kernel density estimation using the fast fourier transform," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 1, pp. 93–99, 1982.

[13] J. Fan and J. S. Marron, "Fast implementations of nonparametric curve estimators," *Journal of computational and graphical statistics*, vol. 3, no. 1, pp. 35–56, 1994.

[14] M. P. Wand, "Fast computation of multivariate kernel estimators," *Journal of Computational and Graphical Statistics*, vol. 3, no. 4, pp. 433–445, 1994.

[15] P. Hall and M. P. Wand, "On the accuracy of binned kernel density estimators," *Journal of Multivariate Analysis*, vol. 56, no. 2, pp. 165–184, 1996.

[16] W. M. Wells, "Efficient synthesis of gaussian filters by cascaded uniform filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 234–239, 1986.

[17] P. Gwosdek, S. Grewenig, A. Bruhn, and J. Weickert, "Theoretical foundations of gaussian convolution by extended box filtering," in *Int. Conf. on Scale Space and Variational Methods in Computer Vision*. Springer, 2011, pp. 447–458.

[18] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed. California Technical Pub, 1999.

[19] F. Ebner, T. Fetzer, L. Köping, M. Grzegorzek, and F. Deinzer, "Multi Sensor 3D Indoor Localisation," in *Indoor Positioning and Indoor Navigation (IPIN), Int. Conf. on*. IEEE, 2015.

[20] F. Ebner, T. Fetzer, F. Deinzer, and M. Grzegorzek, "On wi-fi model optimizations for smartphone-based indoor localization," *ISPRS International Journal of Geo-Information*, vol. 6, no. 8, 2017.

[21] T. Fetzer, F. Ebner, F. Deinzer, and M. Grzegorzek, "Recovering from sample impoverishment in context of indoor localisation," in *Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN)*, 2017, pp. 1–8.