Fast Kernel Density Estimation using Gaussian Filter Approximation

Indoor Navigation Research Group

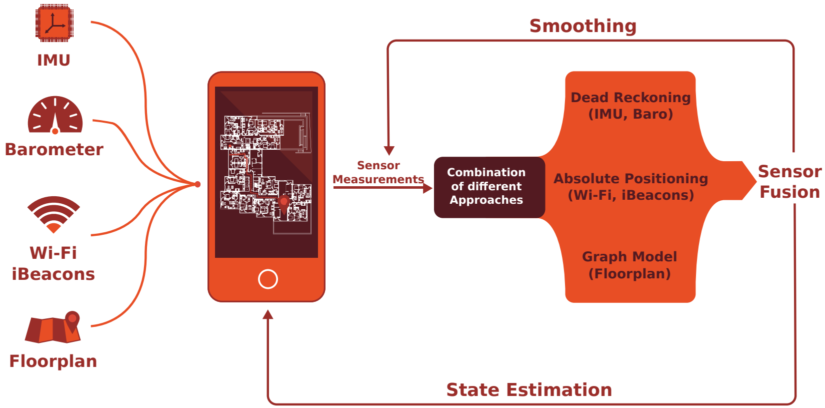
12 July 2018

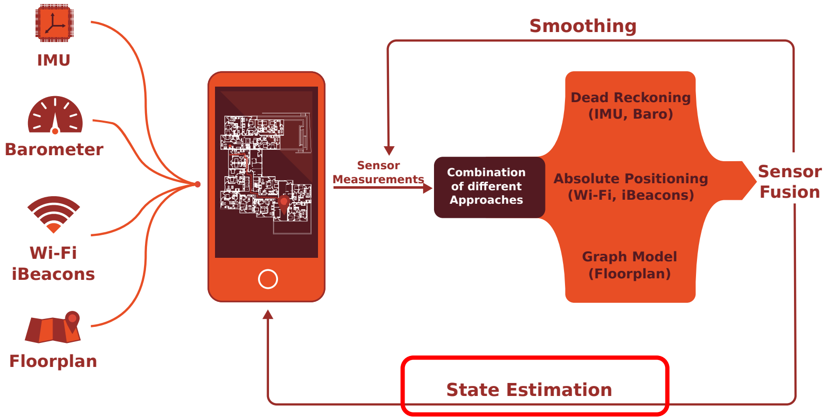
M. Bullmann[1], T. Fetzer[1], F. Ebner[1], F. Deinzer[1], and M. Grzegorzek[2]

[1] University of Applied Sciences Würzburg-Schweinfurt
[2] University of Siegen - Pattern Recognition Group

BACKGROUND

INDOOR LOCALISATION




IMU
Barometer
Wi-Fi
iBeacons
Floorplan
Sensor
Measurements
Smoothing
Combination
of different
Approaches
Dead Reckoning
(IMU, Baro)
Absolute Positioning
(Wi-Fi, iBeacons)
Graph Model
(Floorplan)
Sensor
Fusion
State Estimation

SYSTEM PROPERTIES

- Particle filter with importance sampling describes posterior as a set of samples
- One particle: $\boldsymbol{p_i} = \langle \boldsymbol{q}_i, w_i \rangle$
- Possible state: $\boldsymbol{q}_i = (x, y, z, \theta, \hat{\rho}_{\text{rel}})^T$

$$\overbrace{x, y, z \in \mathbf{R}}^{\text{position}}, \quad \overbrace{\theta \in \mathbf{R}}^{\text{heading}}, \quad \overbrace{\hat{\rho}_{\text{rel}}}^{\text{rel. pressure}}$$

- Constrained computational environment (Smartphone + 500ms time frames)

SYSTEM PROPERTIES

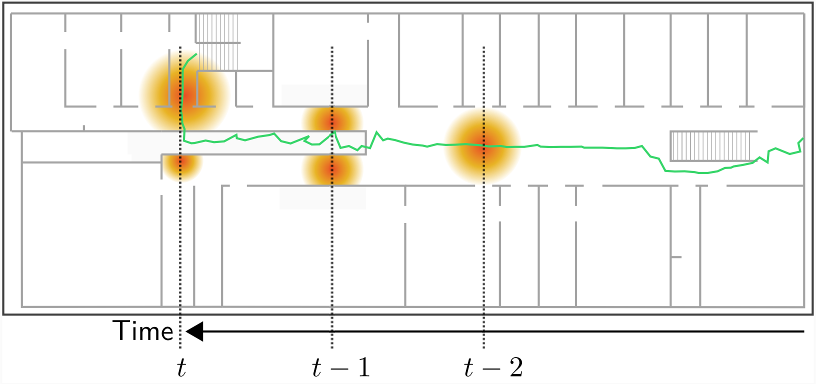
- Particle filter with importance sampling describes posterior as a set of samples
- One particle: $\boldsymbol{p_i} = \langle \boldsymbol{q}_i, w_i \rangle$
- Possible state: $\boldsymbol{q}_i = (x, y, z, \theta, \hat{\rho}_{\text{rel}})^T$

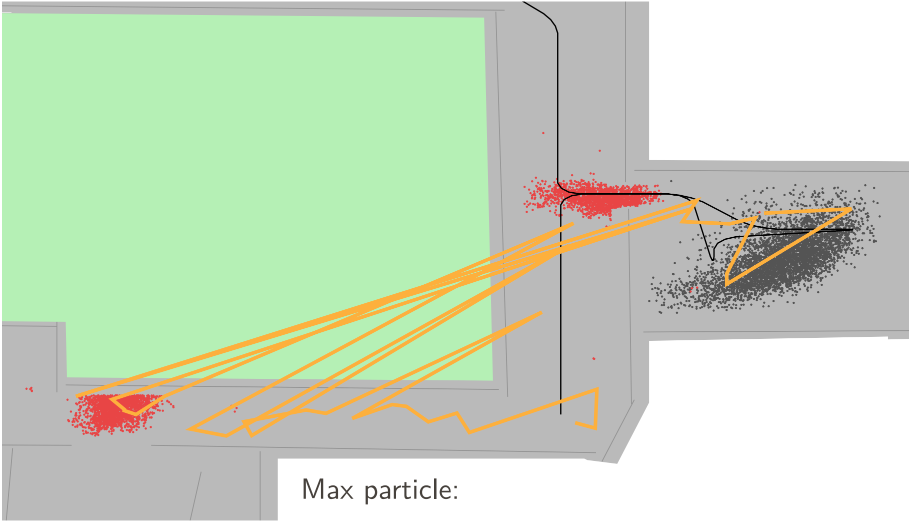
$$\boxed{\overbrace{x, y, z \in \mathbf{R},}^{\text{position}}} \quad \overbrace{\theta \in \mathbf{R},}^{\text{heading}} \quad \overbrace{\hat{\rho}_{\text{rel}}}^{\text{rel. pressure}}$$

- Constrained computational environment (Smartphone + 500ms time frames)

OUR PROBLEM

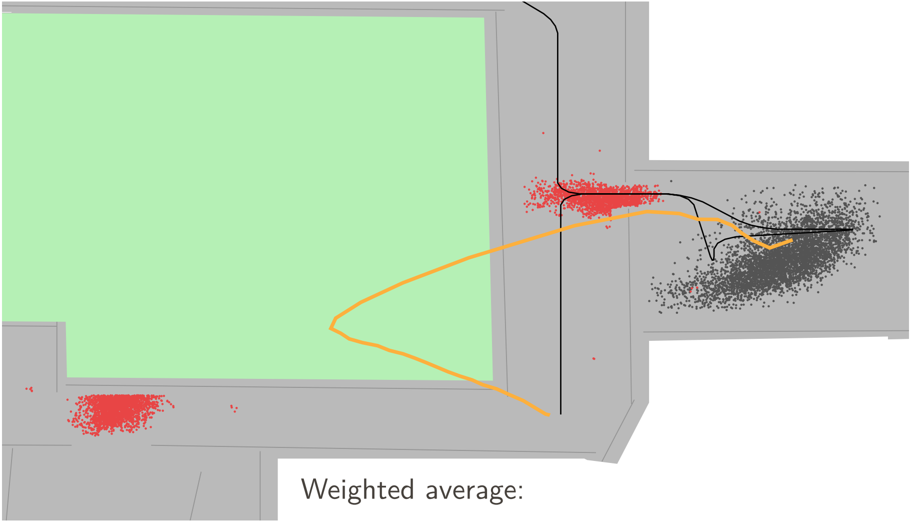
Increased error due to multi modalities





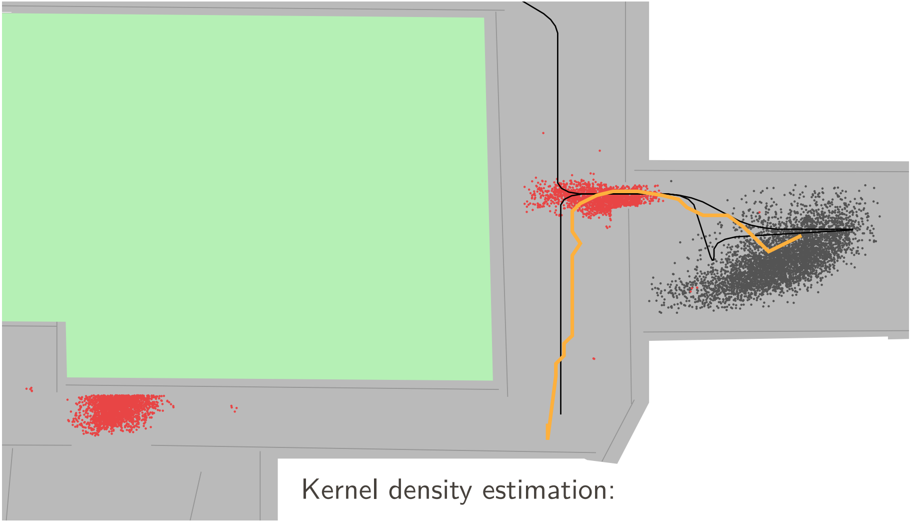
Max particle:

$$\hat{\boldsymbol{q}} := \max_{\boldsymbol{q}_i}\{w_i\}$$



Weighted average:

$$\hat{\boldsymbol{q}} := \frac{1}{W} \sum_{i=1}^{N} w_i \boldsymbol{q}_i \text{ with } W = \sum_{i=1}^{N} w_i$$



Kernel density estimation:

$$\hat{\boldsymbol{q}} := \arg\max_{\boldsymbol{q}_i} \hat{f}(\boldsymbol{q}_i)$$

KERNEL DENSITY ESTIMATION

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

with Gaussian kernel

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

and kernel bandwidth

$$h \in \mathbf{R}^+$$

$\Rightarrow$ Complexity: $\mathcal{O}(MN)$

APPROXIMATION METHOD

BINNED KERNEL DENSITY ESTIMATION

Approximation of KDE assuming an equidistant grid with G grid points

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

where the count at g_j is given with

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

and r_j is a binning rule (simple or linear)

$\Rightarrow$ Complexity: $\mathcal{O}\left(G^2\right)$

BINNED KERNEL DENSITY ESTIMATION

Binned KDE has a discrete convolution structure

$$\tilde{f}(g_x) = \frac{1}{W} \sum_{j=1}^{G} \frac{C_j}{h} K\left(\frac{g_x - g_j}{h}\right) \qquad (f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

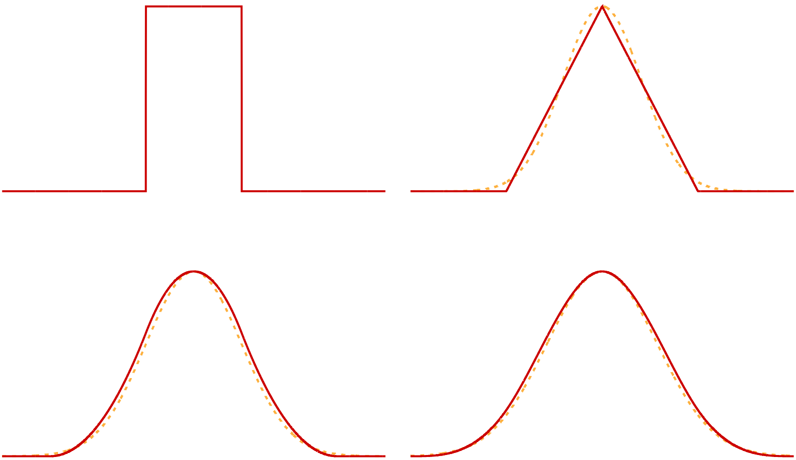
Using a Gaussian kernel: BKDE $\approx$ Gaussian filter

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

$\Rightarrow$ fast approximation using recursive box filter

BOX FILTER APPROXIMATION

Given by central limit theorem, iterated box filters converge to Gaussian filter



BOX FILTER APPROXIMATION

Naive calculation $\mathcal{O}(NL)$:

$$y[i] = \frac{1}{2l+1} \sum_{j=-l}^{l} x[i+j]$$

Radius: $l = \frac{L-1}{2}$

Recursive calculation $\mathcal{O}(N)$:

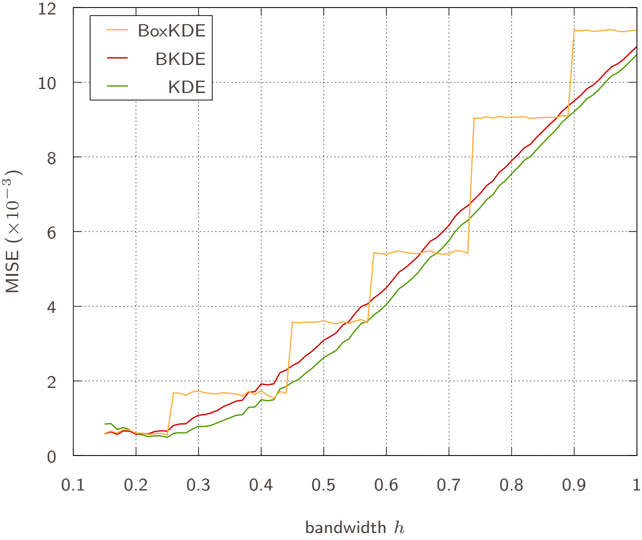
$$y[i] = y[i-1] + x[i+l] - x[i-l-1]$$

Mapping bandwidth to box filter radius:

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

EXPERIMENTS

APPROXIMATION ERROR



Average error ($\times 10^{-3}$):

$$\overline{\text{BoxKDE}} = 5.14$$

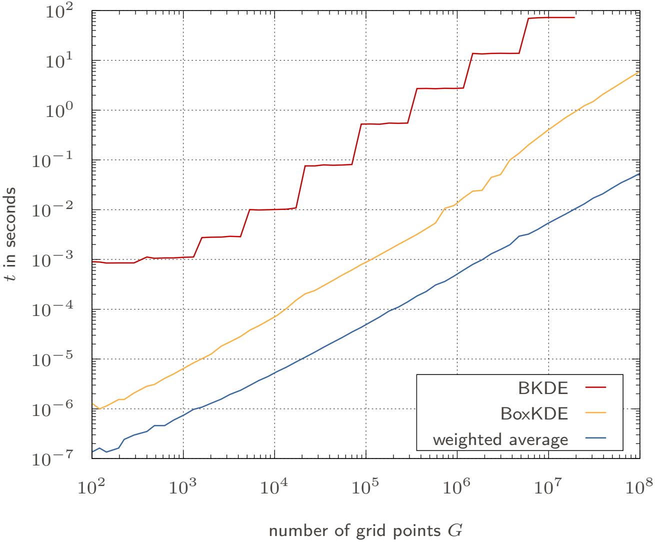
$$\overline{\text{BKDE}} = 4.73$$

$$\overline{\text{KDE}} = 4.40$$

PERFORMANCE

- Hardware
  - Intel Core i5-7600K CPU with 4.2 GHz
  - 16 GB main memory
- Algorithms
  - C++ implementation of our BoxKDE approximation
  - R language package `ks`
    (FFT-based BKDE; optimized in C)
  - C++ implementation of weighted average
- Input
  - Number of grid points $G \in [10^2, 10^8]$

PERFORMANCE



Average time:

$$\bar{t}_{\text{BKDE}} = 10.10\,\text{s}$$
$$\bar{t}_{\text{BoxKDE}} = 0.4092\,\text{s}$$
$$\bar{t}_{\text{WA}} = 0.0043\,\text{s}$$

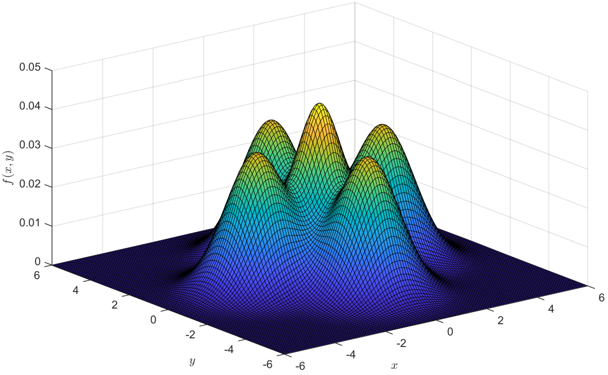
SUMMARY

SUMMARY

- Novel rapid approximation scheme for KDE
- Key idea is to interpret KDE as filter problem
- Assuming a Gaussian kernel, box filter approximation can be used
- Sufficiently fast for our real time application
- Acceptable small error

THANK YOU

APPROXIMATION ERROR - GROUND TRUTH



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

$N = 5000$

Grid: 30×30