Density-Preserving Hierarchical EM Algorithm: Simplifying Gaussian Mixture Models for Approximate Inference

Lei Yu, Tianyu Yang, and Antoni B. Chan

Abstract—We propose an algorithm for simplifying a finite mixture model into a reduced mixture model with fewer mixture components. The reduced model is obtained by maximizing a variational lower bound of the expected log-likelihood of a set of virtual samples. We develop three applications for our mixture simplification algorithm: recursive Bayesian filtering using Gaussian mixture model posteriors, KDE mixture reduction, and belief propagation without sampling. For recursive Bayesian filtering, we propose an efficient algorithm for approximating an arbitrary likelihood function as a sum of scaled Gaussian. Experiments on synthetic data, human location modeling, visual tracking, and vehicle self-localization show that our algorithm can be widely used for probabilistic data analysis, and is more accurate than other mixture simplification methods.

Index Terms—density simplification, likelihood approximation, Gaussian mixture model, recursive Bayesian filtering.

1 Introduction

Finite mixture models are commonly used tools for data analysis and modelling as they are universal approximators for any continuous probability density [1–3]. A typical finite mixture model is given as a weighted combination of components, $f(x) = \Sigma_{i=1}^{K_b} \pi_i f_i(x)$, where $f_i(x)$ is the density of the i-th component, π_i is its corresponding weight, and K_b is the number of components. By choosing appropriate representative components f_i, the mixture model provides more flexibility to model the local variation of observed data [2]. Normally, a mixture model with a large number of components more accurately characterizes irregular and multimodal distributed data. Yet, the number of components cannot be too large as it increases the computation and time required in real applications.

This is especially evident when mixture models are used in Bayesian inference. For example, in non-parametric belief propagation, when the current beliefs are represented as mixture models, the number of mixture components increases exponentially as message passing iterates. Similarly, for recursive Bayesian filtering, when the posterior density is a mixture model, the number of components will increase exponentially if the likelihood function is also a mixture. On the other hand, a kernel density estimator (KDE) [4] can be considered as a mixture model fitted with equal component weights $\pi_i = 1/N$ and the number of components K is the same as the number of data points N. The likelihood computation of a new test data will be time consuming when N is extremely large, which is common with the advent of social networks and cloud computing.

In order to prevent the number of components involved in the inference process from increasing dramatically, some intermediate probabilities can be approximated with randomly selected samples according to certain distributions, such as particle filtering [5–15]. To accelerate the likelihood computation of a new test point with KDE, a kd-Tree approximation can be used that only considers points within the test point's kd-Tree cells [16]. Another commonly used approach is to simplify the mixture model $f(x)$ to $g(x) = \Sigma_{j=1}^{K_r} \pi_j g_j(x)$ with $K_r \ll K_b$, such that the primary structure of the base density $f(x)$ is preserved (e.g., see Fig. 1(b)). In this paper, we focus on directly simplifying a mixture model by reducing the number of components, while keeping the same shape.

In general, there are two approaches to obtain a reduced (simplified) mixture model. The first approach performs density estimation on samples from the base mixture model, such as EM [17], unsupervised fitting according to Minimum Message Length(MML) criterion [18], and Bayesian non-parametric methods [19]. The second approach groups the components of the original mixture into clusters, and then estimates a representative density for each cluster to obtain the reduced mixture model. Clustering is achieved by minimizing a distance (or dissimilarity) between the original (base) and reduced mixture models, with methods differing on the specific distance used, e.g., KL divergence (KLD) [20, 21], differential entropy [22], Bregman divergences [23], expected log-likelihood (EL) [24], or L_2 norm [25]. KLD and EL have no closed-form solution for mixture models, and hence approximations are needed (e.g., variational approximation [21], or an altered formulation [24]).

In this paper, we follow the clustering approach and propose a novel density simplification algorithm that can well preserve the original mixture distribution by directly grouping the base probability densities. Our work is inspired by the hierarchical EM (HEM) algorithm [24], which clusters components of a Gaussian mixture model (GMM) into groups and learns a novel Gaussian cluster center to represent each group, forming a new reduced GMM. Although it works well for clustering, the reduced mixture model produced by HEM does not preserve the structure of

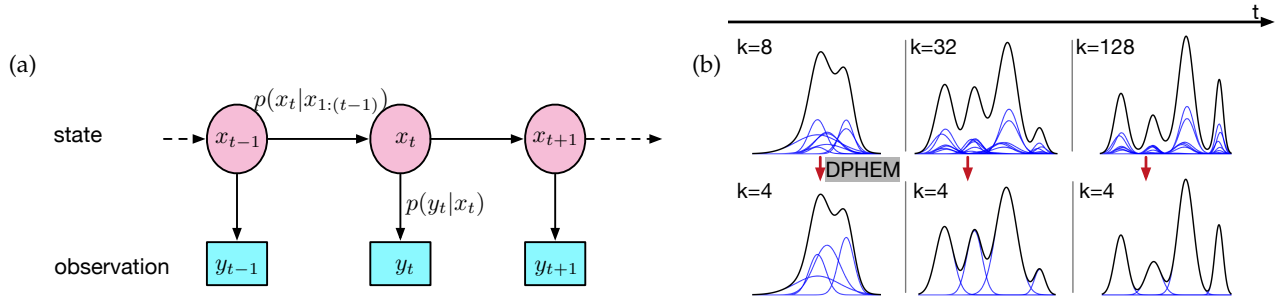


Fig. 1. (a) A typical framework for a first-order Markov model, where x_t is the state at time t (e.g., object location) and y_t is the observation at time t (e.g., image frame). (b). An illustration of model size growing with time. Simplification is applied to prevent it from growing too large.

the original mixture, due to its multiple instance learning property [26], which is derived from an altered formulation that forces consistency in cluster assignments. To address this problem, we derive a new *density-preserving* HEM algorithm from first principles, which is based on a variational approximation to the expected log-likelihood between two mixture models. The contributions of this paper are 4-fold: 1) we propose a novel density-preserving HEM (DPHEM) algorithm that estimates a reduced mixture that well preserves the structure of the base mixture model; 2) we develop a complete framework for recursive Bayesian filtering where the posteriors are represented as GMMs and the likelihood functions are approximated as sums of scaled Gaussians (SSG), and the complexity of the GMM posterior is controlled by DPHEM; 3) to facilitate our recursive Bayesian filtering framework, we propose an algorithm to approximate an arbitrary likelihood function as an SSG; 4) we test DPHEM on a variety of applications, including visual tracking and vehicle localization (both using recursive Bayesian filtering), non-parametric belief propagation, and human location modeling (KDE mixture reduction), and demonstrate the superiority of DPHEM to other density simplification methods.

The remainder of this paper is organized as follows. We briefly review related works in Section 2. We derive the density-preserving HEM algorithm and compare with related methods in Section 3. More details of the derivation and proofs can be found in the Appendix. Next, in Section 4, we present two applications of DPHEM. The likelihood approximation algorithm proposed to unify the recursive Bayesian inference for arbitrary likelihood functions is presented in Section 4.1.1. Four experiments to verify DPHEM on these applications are presented in Section 5.

2 Related Work

Density simplification (or density reduction) is used in recursive Bayesian filtering in order to control the complexity of the posterior distribution. [27] uses a piecewise GMM for the posterior distribution, and in each step matches the closest components between the prior and posterior to prevent an increase in the number of components. In visual tracking, particle filtering [28] represents the posterior of the object state as a set of weighted samples. In [29, 30], a GMM is constructed using samples from the base distribution, where an upper bound constraint is imposed on the covariance of new components. In non-parametric belief propagation, sampling methods are normally applied to make the inference tractable [31–37]. In approximate message passing (AMP), tractable message passing is achieved by forming simplified mixture models by sampling from the base likelihood function or from the application-specific data [38]. As an alternative to sampling methods, [21] proposed an algorithm (denoted as VKL) to remove components from a GMM by minimizing a variational approximation of the Kullback-Leibler (KL) divergence between the original GMM and the simplified model. While VKL solves this problem to some degree, important components are sometimes lost because the component with smallest weight is removed in each iteration. Other applications include pose estimation [37], background subtraction in video [39], image denoising [40], stereo matching and optical flow estimation [41] and so on.

Related to density simplification is the task of density clustering, where the goal is to group probability densities into clusters and to estimate representative densities for each cluster (i.e., the cluster "centers"). While the collection of representative densities could be interpreted as a mixture model, the clustering methods do not necessarily guarantee that the new model well preserves the shape of the original density, as they mainly focus on clustering. [23] introduces a clustering algorithm based on Bregman divergences, and using discrete KL divergence yields an algorithm for clustering multinomials. When the Bregman divergence consists of the sum of the Mahalonobis distance and the Burg matrix divergence, the result is a clustering algorithm for Gaussians [22] with hard assignments. The original HEM [24] is a soft-clustering algorithm, which is a generalization of Bregman clustering [22] - [22] is a special case when the number of virtual samples grows to infinity. Similarly, [20] minimizes the weighted sum of the KL divergence between the cluster center and each probability distribution, and uses a hard-clustering approach. Our proposed DPHEM algorithm is a soft clustering algorithm, which is a generalization of [20]. [25] performs the clustering by minimizing an upper bound of the approximation error which is measured with the L_2 norm between the base and simplified mixture model. Since probability densities lie in a non-linear manifold, an alternative approach is to use spectral clustering [42] or non-linear dimensionality reduction and K-means clustering [43]. While the result is a grouping of densities into clusters, [42, 43] are not able to generate novel cluster centers due to the pre-image and out-of-sample limitations of kernel methods. One solution [43] is to use the medoid (the closest sample to the mean in the embedding) as the cluster center, but this is suboptimal for hierarchical clustering and estimation. [44] applies a variational Bayes framework for merging Gaussian mixture components within the framework of [24]. Hierarchies of mixtures, obtained by recursively applying HEM to a mixture model, have been used for image/video

annotation [45], efficient indexing for image retrieval [46], and codebook learning [45, 47] and inference [48].

Our approach (DP-HEM) is most similar to the VKL [21] and HEM algorithms [24]. Similar to HEM, we formulate mixture simplification as maximizing the expected log-likelihood (EL) of the reduced mixture. However, in contrast to HEM, we do not force a consistency requirement that a block of virtual samples from a given base component must be assigned to the same reduced mixture component. The result is a DP-HEM algorithm that preserves the shape of the density. Compared to VKL, our DP-HEM maximizes a variational lower bound of the EL between the base and reduced mixtures, while VKL minimizes a variational upper bound of the KL divergence (KLD). Ideally, there should be an equivalence between maximizing the EL and minimizing the KLD, since the negative entropy term of KLD does not affect its minimization. However, due to the variational approximation, the variational KLD forms a looser bound than that of the variational EL. The result is that our DP-HEM yields more accurate reduced mixtures. The detailed differences between VKL, HEM, and our DPHEM are discussed in Section 3.3.

3 Density-preserving HEM algorithm

In this section, we derive a hierarchical EM algorithm that takes an input mixture model (e.g., GMM) and estimates an equivalent mixture model with fewer number of components. Formally, let $\Theta^{(b)}$ represent a "base" (input) mixture density with K_b components. The i-th component has density $p(y|\theta_i^{(b)})$ with parameters $\theta_i^{(b)}$ and prior probability $\pi_i^{(b)}$. The likelihood of an observation $y \sim \Theta^{(b)}$ is given by

$$p(y|\Theta^{(b)}) = \sum_{i=1}^{K_b} \pi_i^{(b)} p(y|\theta_i^{(b)}). \tag{1}$$

Our goal is to simplify the base model $\Theta^{(b)}$ to a "reduced" mixture $\Theta^{(r)}$ with much fewer components $K_r \ll K_b$,

$$p(y|\Theta^{(r)}) = \sum_{j=1}^{K_r} \pi_j^{(r)} p(y|\theta_j^{(r)}). \tag{2}$$

Note that we will always use i and j to index the base and reduced mixture components, respectively.

One possible solution to estimate $\Theta^{(r)}$ is to directly sample from $\Theta^{(b)}$ and then estimate $\Theta^{(r)}$ with any needed number of components by EM algorithm. However, this would be inefficient when handling large-scale high-dimensional data. Instead, we take our inspiration from HEM [24], where a hierarchical clustering of mixture models is obtained directly from the parameters of the base mixture components using a set of *virtual samples*.

We define a set of N virtual samples, $Y = \{y_1, y_2, \cdots, y_N\}$, which are i.i.d. samples from the base mixture model, $y_n \sim \Theta^{(b)}$. The reduced model $\Theta^{(r)}$ can then be obtained by maximizing the *expected* log-likelihood of the reduced model $\Theta^{(r)}$ with respect to the virtual samples[1],

$$\begin{aligned}\mathcal{J}(\Theta^{(r)}) &= \mathbb{E}_{Y|\Theta^{(b)}}[\log p(Y|\Theta^{(r)})] \\ &= \sum_i \pi_i^{(b)} \mathbb{E}_{Y|\theta_i^{(b)}}[\log p(Y|\Theta^{(r)})].\end{aligned} \tag{3}$$

1. See Appendix A for the detailed derivation.

However, maximization of (3) is intractable because the expected log-likelihood of a mixture model $\mathbb{E}_{Y|\theta_i^{(b)}}[\log p(Y|\Theta^{(r)})]$ cannot be calculated in closed-form. Here we adopt a variational approximation method for this optimization.

3.1 Variational approximation

We use a variational perspective of the EM algorithm [49–51], which treats the E- and M-steps both as maximization processes. We start from a variational lower bound of the log-likelihood of a mixture model [52, 53]

$$\log p(Y|\Theta^{(r)}) \geq \max_{z_{ij}} \sum_j z_{ij} \log \frac{\pi_j^{(r)} p(Y|\theta_j^{(r)})}{z_{ij}}, \tag{4}$$

where z_{ij} are the variational parameters and $\sum_{j=1}^{K_r} z_{ij} = 1$. Taking the expectation of (4) and using Jensen's inequality,

$$\begin{aligned}&\mathbb{E}_{Y|\theta_i^{(b)}}[\log p(Y|\Theta^{(r)})] \\ &\geq \max_{z_{ij}} \sum_j z_{ij} \left\{ \log \frac{\pi_j^{(r)}}{z_{ij}} + \mathbb{E}_{Y|\theta_i^{(b)}}[\log p(Y|\theta_j^{(r)})] \right\}.\end{aligned} \tag{5}$$

The inner-expectation in (5) is obtained by noting that Y is a set of i.i.d. samples,

$$\begin{aligned}&\mathbb{E}_{Y|\theta_i^{(b)}}[\log p(Y|\theta_j^{(r)})] = \mathbb{E}_{Y|\theta_i^{(b)}}\left[\sum_{n=1}^{N} \log p(y_n|\theta_j^{(r)})\right] \\ &= \sum_{n=1}^{N} \mathbb{E}_{Y|\theta_i^{(b)}}[\log p(y_n|\theta_j^{(r)})] = N\mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})].\end{aligned} \tag{6}$$

Finally, substituting (6) and (5) into (3), we obtain the variational lower bound of the expected log-likelihood,

$$\begin{aligned}&\mathcal{J}_{DP}(\Theta^{(r)}) \\ &= \max_{z_{ij}} \sum_i \sum_j \pi_i^{(b)} z_{ij} \left\{ \log \frac{\pi_j^{(r)}}{z_{ij}} + N\mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})] \right\} \\ &\leq \mathbb{E}_{Y|\Theta^{(b)}}[\log p(Y|\Theta^{(r)})].\end{aligned} \tag{7}$$

The variational parameters z_{ij} can be interpreted as an assignment of the virtual samples generated from the ith base component to the jth reduced component. The variational lower bound in (7) is maximized by iterating between maximizing w.r.t. the assignments z_{ij} and the reduced mixture parameters $\Theta^{(r)}$.

3.2 DPHEM for GMMs

Next we present the DPHEM algorithm for the specific case of Gaussian mixture models (see Appendix B for derivation details). Let the base and reduced mixture components be Gaussians, $p(y|\theta_i^{(b)}) = \mathcal{N}(y|\mu_i^{(b)}, \Sigma_i^{(b)})$ and $p(y|\theta_j^{(r)}) = \mathcal{N}(y|\mu_j^{(r)}, \Sigma_j^{(r)})$, where $\mathcal{N}(y|\mu, \Sigma)$ is a multivariate Gaussian density with mean μ and covariance Σ. The parameters of the base and reduced models are $\Theta^{(b)} = \{\pi_i^{(b)}, \mu_i^{(b)}, \Sigma_i^{(b)}\}_{i=1}^{K_b}$ and $\Theta^{(r)} = \{\pi_j^{(r)}, \mu_j^{(r)}, \Sigma_j^{(r)}\}_{j=1}^{K_r}$.

3.2.1 E-step

Maximizing the lower bound in (7) with respect to variational parameter yields

$$\hat{z}_{ij} = \frac{\pi_j^{(r)} \exp(N\mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})])}{\sum_{j'=1}^{K_r} \pi_{j'}^{(r)} \exp(N\mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_{j'}^{(r)})])}, \tag{8}$$

where the expected log-Gaussian between $\theta_i^{(b)}$ and $\theta_j^{(b)}$ is

$$\mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})] = \log \mathcal{N}(\mu_i^{(b)}|\mu_j^{(r)}, \Sigma_j^{(r)}) - \frac{1}{2}\mathrm{tr}\{(\Sigma_j^{(r)})^{-1}\Sigma_i^{(b)}\}. \quad (9)$$

In (8), the number of virtual samples N controls the "peakiness" of the assignment variable $\hat{z}_{ij}$. Letting $N \to \infty$ results in hard assignments, while $0 < N < \infty$ yields soft assignments. In our experiments, we set N as a multiple of the number of base components, $N = \alpha K_b$, where $\alpha = 10$.

3.2.2 M-step

Given the optimal variational parameters, the lower bound is maximized w.r.t. to the model parameters $\Theta^{(r)} = \{\pi_j^{(r)}, \mu_j^{(r)}, \Sigma_j^{(r)}\}$, yielding the parameter updates:

$$\hat{\pi}_j^{(r)} = \sum_{i=1}^{K_b} \hat{z}_{ij}\pi_i^{(b)}, \qquad \hat{\mu}_j^{(r)} = \frac{1}{\hat{\pi}_j^{(r)}} \sum_{i=1}^{K_b} \hat{z}_{ij}\pi_i^{(b)}\mu_i^{(b)}, \quad (10)$$

$$\hat{\Sigma}_j^{(r)} = \frac{1}{\hat{\pi}_j^{(r)}} \sum_{i=1}^{K_b} \hat{z}_{ij}\pi_i^{(b)}[\Sigma_i^{(b)} + (\mu_i^{(b)} - \hat{\mu}_j^{(r)})(\mu_i^{(b)} - \hat{\mu}_j^{(r)})^T]. \quad (11)$$

3.3 Comparison to related works

Due to different formulations, our DPHEM has a few key differences from HEM [24] and VKL [21]. While the update equations for the component means and covariances are the same, the three algorithms differ in their assignment variables and component weights. Details can be found in Table 1. Since L2U [25] uses hard clustering, which is inherently different from the soft clustering used by HEM, VKL and DPHEM, the update formula for [25] is not presented in Table 1. We next discuss detailed comparisons between DPHEM and the three methods: HEM, VKL, and L2U.

TABLE 1
Comparison of DPHEM with related works. To reduce clutter, here we use $E_{ij} = \mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})]$.

	assignment variable	component weight
HEM [24]	$\hat{z}_{ij} \propto \pi_j^{(r)} \exp(\pi_i^{(b)} N E_{ij})$	$\hat{\pi}_j^{(r)} = \frac{1}{K_b}\sum_i \hat{z}_{ij}$
DPHEM	$\hat{z}_{ij} \propto \pi_j^{(r)} \exp(N E_{ij})$	$\hat{\pi}_j^{(r)} = \sum_i \hat{z}_{ij}\pi_i^{(b)}$
VKL [21]	$\hat{z}_{ij} \propto \pi_j^{(r)}\hat{\zeta}_{ij} \exp(E_{ij})$ $\hat{\zeta}_{ij} = \hat{z}_{ij}\pi_i^{(b)}/(\sum_{i'} \hat{z}_{i'j}\pi_{i'}^{(b)})$	$\hat{\pi}_j^{(r)} = \sum_i \hat{z}_{ij}\pi_i^{(b)}$

3.3.1 Comparison with HEM

The original HEM algorithm [24] forms a set of virtual sample blocks $\{Y_1, \cdots Y_{K_b}\}$, where each block $Y_i = \{y_{i,n}\}_{n=1}^{N_i}$ contains $N_i = \pi_i^{(b)}N$ samples from one base component, $y_{i,n} \sim \theta_i^{(b)}$. The reason for this construction is so that virtual samples from the same base component will be assigned to the same reduced mixture component, thus preserving a consistent hierarchy. The reduced model is found by applying the EM algorithm to maximize the log-likelihood of the sample blocks, $\sum_{i=1}^{K_b} \log p(Y_i|\Theta^{(r)})$, and applying the law of large numbers to replace Y_i with its expectation. An equivalent formulation, shown in [54], is to maximize a variational lower bound on the expected log-likelihood (EL) of the virtual sample blocks under each component,

$$\mathcal{J}_{HEM}(\Theta^{(r)}) \leq \sum_{i=1}^{K_b} \mathbb{E}_{Y_i|\Theta_i^{(b)}}[\log p(Y_i|\Theta^{(r)})]. \quad (12)$$

In contrast, our DPHEM does not form sample blocks, but instead assumes the entire sample set Y contains i.i.d. samples from the full base mixture $\Theta^{(b)}$. DPHEM then maximizes a variational lower bound on the expected log-likelihood, where the expectation is w.r.t. the *whole* base mixture, as in (7).

The different formulations of HEM and DPHEM lead to algorithms with different properties. For HEM, the reduced component weight $\hat{\pi}_j^{(r)} = \frac{1}{K_b}\sum_i \hat{z}_{ij}$ is only determined by the number of assigned base mixture components, and ignores the weight $\pi_i^{(b)}$ of the base component. Hence, HEM will promote commonly occurring mixture components, while suppressing infrequent outlier components, regardless of the actual mixture weights. This property is useful for multiple instance-learning, e.g., where the goal is to learn the commonly occurring features existing in the noisy training datasets [26]. However, HEM will perform poorly when the goal is to estimate a simplified mixture model that best represents the base density structure. In contrast, our DPHEM includes the base component weights when computing $\hat{\pi}_j^{(r)}$, resulting in better representations of the base mixture. An example is presented in Fig. 2a. Note that HEM and DPHEM are equivalent when the base component weights are uniform, $\pi_i^{(b)} = \frac{1}{K_b}$.

3.3.2 Comparison with VKL

VKL [21] estimates the reduced mixture $\Theta^{(r)}$ by minimizing the KL divergence (KLD) between the base mixture and the reduced mixture,

$$\hat{\Theta}^{(r)} = \underset{\Theta^{(r)}}{\mathrm{argmin}}\, D(\Theta^{(b)}\|\Theta^{(r)}), \quad (13)$$

where the KLD between the two mixtures is

$$D(\Theta^{(b)}\|\Theta^{(r)}) = \int p(y|\Theta^{(b)}) \log \frac{p(y|\Theta^{(b)})}{p(y|\Theta^{(r)})} dy \quad (14)$$

$$= \mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(b)})] - \mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(r)})]. \quad (15)$$

When the KLD and EL can be exactly computed, there is an equivalence between formulations,

$$\hat{\Theta}^{(r)} = \underset{\Theta^{(r)}}{\mathrm{argmin}}\, D(\Theta^{(b)}\|\Theta^{(r)}) \quad (16)$$

$$= \underset{\Theta^{(r)}}{\mathrm{argmax}}\, \mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(r)})], \quad (17)$$

since the first term in (15) is a constant w.r.t. $\Theta^{(r)}$. However, as the two densities are mixture models, [21] minimizes a variational upper-bound of the KLD,

$$\mathcal{J}_{KL}(\Theta^{(r)}) = \min_{z_{ij},\zeta_{ij}} \sum_{i,j} \pi_i^{(b)} z_{ij}(\log \frac{\pi_i^{(b)} z_{ij}}{\pi_j^{(r)}\zeta_{ij}} + D(\theta_i^{(b)}\|\theta_j^{(r)})) \geq D(\Theta^{(b)}\|\Theta^{(r)}), \quad (18)$$

where $z_{ij} \geq 0$ and $\zeta_{ij} \geq 0$ are two sets of variational parameters [55][2], which satisfy the constraints $\sum_j z_{ij} = 1, \forall i$ and $\sum_i \zeta_{ij} = 1, \forall j$. Substituting (15) into (18), we have

$$\mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(b)})] - \mathcal{J}_{KL}(\Theta^{(r)}) \leq \mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(r)})]. \quad (19)$$

Dropping constant terms on the LHS of (19), we obtain an equivalent maximization problem for VKL (see Appendix

2. To make the VKL formulation comparable with HEM/DPHEM, we change the notation from [55] using $\phi_{j|i} = \pi_i^{(b)} z_{i,j}$ and $\psi_{i|j} = \pi_j^{(r)}\zeta_{i,j}$.

C),

$$\tilde{\mathcal{J}}_{KL}(\Theta^{(r)}) = \max_{z_{ij},\zeta_{ij}} \sum_{i,j} \pi_i^{(b)} z_{ij} (\log \frac{\pi_j^{(r)} \zeta_{ij}}{z_{ij}} + \mathbb{E}_{y|\theta_i^{(b)}}[\log p(y|\theta_j^{(r)})]).$$

The relationship between HEM and VKL can be expressed by the following theorem (see Appendix C for proof):

Theorem 1. *DPHEM ($N = 1$) obtains a tighter lower-bound than VKL to the expected log-likelihood,*

$$\tilde{\mathcal{J}}_{KL}(\Theta^{(r)}) \leq \mathcal{J}_{DP}(\Theta^{(r)}) \leq \mathbb{E}_{y|\Theta^{(b)}}[\log p(y|\Theta^{(r)})]. \quad (20)$$

VKL is equivalent to DPHEM, $\tilde{\mathcal{J}}_{KL}(\Theta^{(r)}) = \mathcal{J}_{DP}(\Theta^{(r)})$, when $\hat{\zeta}_{ij} = 1, \forall(i,j)$ where $\hat{z}_{ij} > 0$.

Both methods are maximizing two different lower bounds of the EL to obtain the reduced mixture model $\Theta^{(r)}$, with DPHEM obtaining a tighter lower-bound than VKL. Note that the two lower bounds are the same when $\hat{\zeta}_{ij} = 1$ for all (i,j) where $z_{ij} > 0$. Since $\sum_i \zeta_{ij} = 1$, this can only occur when each reduced component $\theta_j^{(r)}$ only has one base component $\theta_i^{(b)}$ assigned to it. For clustering and density simplification, this is not possible since $K_r < K_b$.

As the DPHEM lower bound is tighter than the VKL lower bound, it is expected that DPHEM will yield a reduced mixture model that better fit the base mixture than VKL. We verify this experimentally in synthetic experiments by comparing real KL divergence between the DPHEM and VKL solutions. Fig. 2b shows an example.

3.3.3 Comparison with L2U

L2U [25] forms the reduced mixture $\Theta^{(r)}$ by minimizing the squared L_2-norm between the density functions of the base and reduced mixture models,

$$\hat{\Theta}^{(r)} = \underset{\Theta^{(r)}}{\operatorname{argmin}} \|p(y|\Theta^{(b)}) - p(y|\Theta^{(r)})\|^2, \quad (21)$$

where the squared L_2-norm between functions is $\|f(y) - g(y)\|^2 = \int (f(y) - g(y))^2 dy$. Since this distance is difficult to optimize directly, [25] first partitions the base mixture components $\{p(y|\theta_i^{(b)})\}$ into disjoint clusters $\{S_1, \cdots, S_{K_r}\}$, and then calculates an upper bound of the squared L_2-norm by applying Cauchy-Schwarz inequality,

$$\mathcal{J}_{L_2}(\Theta^{(r)}) = K_r \sum_j^{K_r} \int (\pi_j^{(r)} p(y|\theta_j^{(r)}) - \sum_{i \in S_j} \pi_i^{(b)} p(y|\theta_i^{(b)}))^2 dy \geq \|p(y|\Theta^{(b)}) - p(y|\Theta^{(r)})\|^2. \quad (22)$$

Then for each cluster j, a good representative $\pi_j^{(r)} p(y|\theta_j^{(r)})$ is found by minimizing the local distance.

Compared to DPHEM, firstly, L2U is a hard clustering algorithm and the result is sensitive to the initial partition of the base mixture components. Secondly, instead of optimizing a "distance" between probability densities (in the log-space), such as EL for HEM and DPHEM, KL divergence for VKL, L2U treats the mixture densities as normal functions and minimizes the L_2-norm. The L_2-norm criteria is good at preserving the regions of the density with large probability, but does not preserve well those regions with small probability. On the other hand, using a log-space distance, such as EL for HEM or KLD for VKL, can better preserve the low-probability regions. An example is shown in Fig. 2c.

We further study the differences between the four methods through experiments in Section 5.1.

4 Applications of DPHEM

We next discuss 2 applications of DPHEM, recursive Bayesian filtering with GMMs, and reducing KDE mixtures.

4.1 Recursive Bayesian filtering with GMMs

Recursive Bayesian filtering estimates the current state of a system using noisy measurements from the past and present. Computer vision applications include object tracking [5, 6, 8–13, 15, 56] and robot localization [21, 57–59]. Fig. 1a shows a typical framework for a 1st-order Markov model, where x_t is the state at time t (e.g., object location) and y_t is the observation at time t (e.g., video frame).

The goal is to obtain the posterior distribution of the current state $p(x_t|y_{1:t})$, conditioned on the seen observations, $y_{1:t} = (y_1, \ldots, y_t)$. The posterior is obtained recursively by first predicting the current state x_t using the previous posterior distribution $p(x_{t-1}|y_{1:t-1})$ and the transition model $p(x_t|x_{t-1})$, and then factoring in the current observation y_t using the observation model $p(y_t|x_t)$,

prediction:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1}) dx_{t-1}, \quad (23)$$

update:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t) p(x_t|y_{1:t-1})}{\int p(y_t|x_t) p(x_t|y_{1:t-1}) dx_t} \quad (24)$$

Assuming that the transition and observation models are Gaussians yields the Kalman filter, which is tractable to compute. For more complex models, one solution is to use a particle filter [5–15], where the posterior is approximated as a set of weighted particles $p(x_t|y_{1:t}) = \sum_m \omega_m \delta(x_t - \hat{x}_m)$, where $(\omega_m, \hat{x}_m)$ is the m-th particle. However, the limited set of samples may not well characterize the true posterior especially when the distribution is heavy-tailed, and errors may accumulate quickly during inference. Increasing the number of particles increases the accuracy of the posterior, but also increases the variance [60] and computational load.

In this paper, we model the posterior distribution $p(x_t|y_{1:t})$ using a GMM,

$$p(x_{t-1}|y_{1:t-1}) = \sum_{i=1}^{N} \pi_i \mathcal{N}(x_{t-1}|\mu_i, \Sigma_i). \quad (25)$$

We assume that the transition model $p(x_t|x_{t-1})$ is a GMM,

$$p(x_t|x_{t-1}) = \sum_{j=1}^{J} \pi_j \mathcal{N}(x_t|Ax_{t-1} + \mu_j, \Sigma_j), \quad (26)$$

where A is a linear transition matrix. When y_t is known, the observation likelihood $p(y_t|x_t)$ is a function of x_t, and we assume it is a sum of scaled Gaussians (SSG)

$$p(y_t|x_t) = \sum_{k=1}^{K} \omega_k \mathcal{N}(x_t|\mu_k, \Sigma_k), \quad (27)$$

where here $\sum_k \omega_k$ is not necessarily 1. The prediction step in (23) can be calculated in closed-form (see Appendix D.2),

$$p(x_t|y_{1:t-1}) = \sum_{j=1}^{J} \sum_{i=1}^{N} \pi_i \pi_j \mathcal{N}(x_t|\mu_{ij}, \Sigma_{ij}), \quad (28)$$

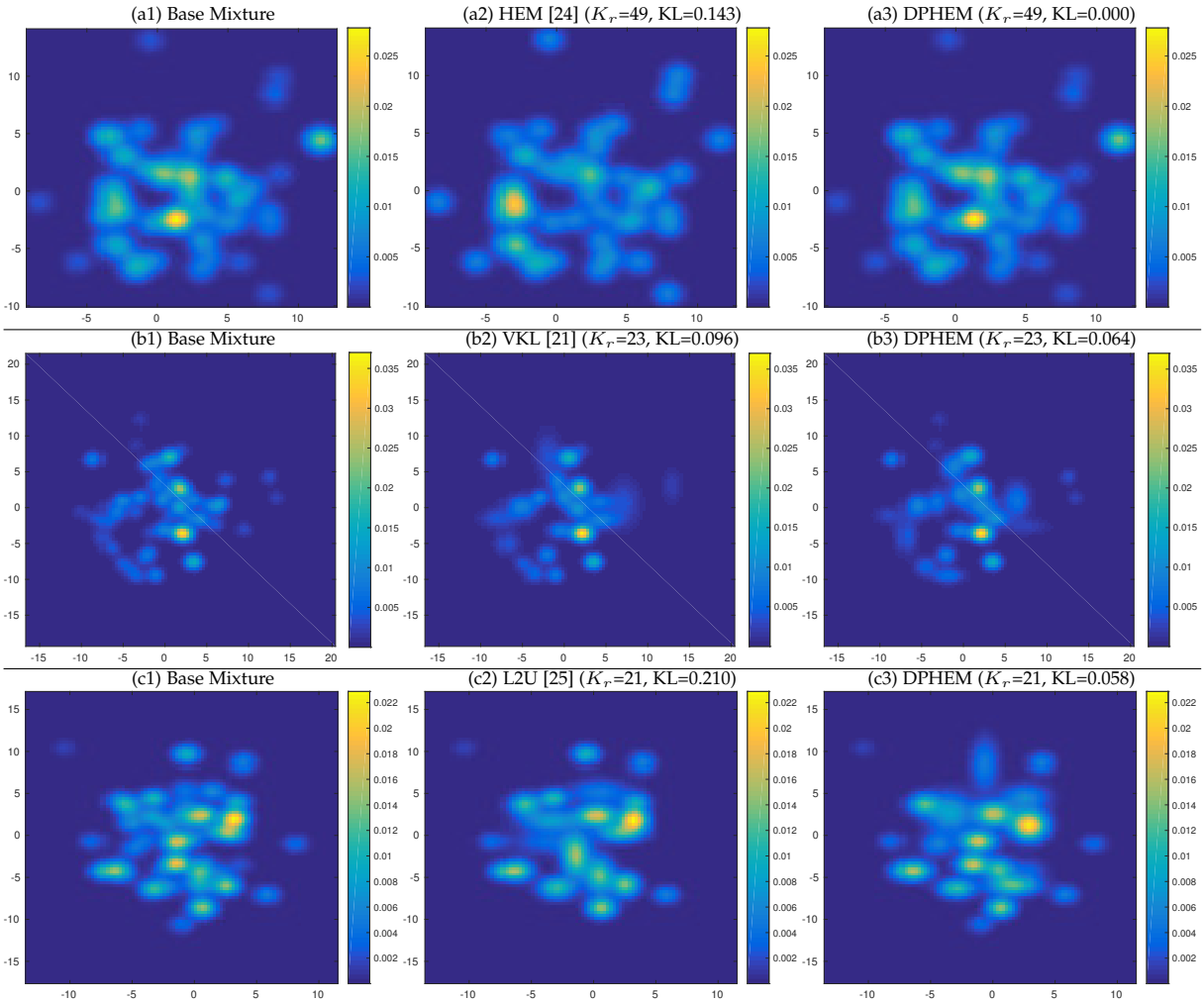


Fig. 2. Examples of reduced mixture models using different approaches. Each row compares DPHEM with a baseline method and shows a typical difference. The base mixture model contains $K_b = 2500$ components, and the reduced mixture contains K_r components. KL is the KL divergence between the base mixture and the reduced mixture.

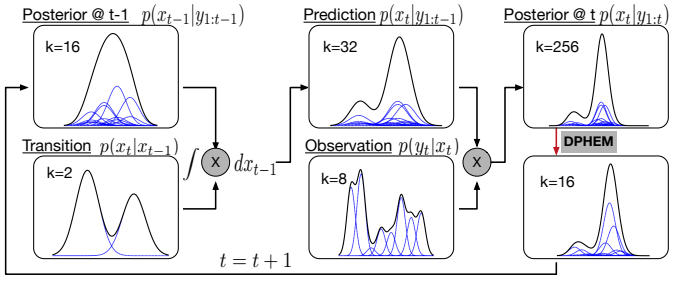


Fig. 3. Recursive Bayesian filtering using GMMs. DPHEM is applied to simplify the posterior GMM.

where $\mu_{ij} = A\mu_i + \mu_j$ and $\Sigma_{ij} = A\Sigma_i A^T + \Sigma_j$. Likewise the updated posterior in (24) can also be calculated in closed-form (see Appendix D.3),

$$p(x_t|y_{1:t}) = \sum_{j=1}^{J}\sum_{i=1}^{N}\sum_{k=1}^{K} \omega_{ijk}\mathcal{N}(x_t|\mu_{ijk}, \Sigma_{ijk}), \tag{29}$$

where $\mu_{ijk} = \Sigma_{ijk}(\Sigma_k^{-1}\mu_k + \Sigma_{ij}^{-1}\mu_{ij})$, and $\Sigma_{ijk} = (\Sigma_k^{-1} + \Sigma_{ij}^{-1})^{-1}$. The weights are $\omega_{ijk} = \frac{\tilde{\omega}_{ijk}}{\sum_{i'j'k'}\tilde{\omega}_{i'j'k'}}$, where $\tilde{\omega}_{ijk} = \pi_i\pi_j\omega_k\mathcal{N}(\mu_k|\mu_{ij}, \Sigma_k + \Sigma_{ij})$. Hence, the posterior is also a GMM, but the number of mixture components will increase exponentially over time. To control the complexity while maintaining an accurate posterior, we apply DPHEM to simplify the posterior when the number of components exceeds a threshold. Fig. 3 illustrates this process.

4.1.1 Likelihood approximation

Our framework assumes that the likelihood function in (27) has the form of a sum of scaled Gaussians (SSG). To facilitate the use of our framework for any likelihood function, we next propose an efficient algorithm for approximating an arbitrary likelihood function as an SSG.

Given a likelihood function $f(x) = p(y|x)$ and a corresponding set of state-likelihood pairs $\mathcal{D} = \{(x_i, p_i)\}_{i=1}^N$, where $p_i = p(y_i|x_i)$, the algorithm computes a lower bound to the points in $\mathcal{D}$ (and hence $f(x)$) by iteratively adding scaled Gaussians to the SSG. The algorithm keeps a list of residuals between $\mathcal{D}$ and the current SSG, denoted as $\mathcal{D}^{(k)} = \{(x_i, r_i)\}_{i=1}^N$. Initially, the set of residual pairs is the set of input pairs, $\mathcal{D}^{(1)} = \mathcal{D}$. In each iteration k, a scaled Gaussian $f^{(k)}(x)$ is found that lower bounds the residuals in $\mathcal{D}^{(k)}$. Then $f^{(k)}(x)$ is added as a component to the SSG, and the next iteration proceeds.

More specifically, in the k-th iteration, first the highest point in $\mathcal{D}^{(k)}$ is found, $m = \text{argmax}_i \log r_i$. Next, the peak of a scaled Gaussian $f^{(k)}$ is anchored on the maximum point,

$$h^{(k)}(x) = -(x - x_m)^T W_k (x - x_m) + \ell_m, \tag{30}$$

$$f^{(k)}(x) = \exp(h^{(k)}(x)), \tag{31}$$

where $\ell_m = \log r_m$ and W_k is the precision matrix of the

Algorithm 1 Likelihood Approximation

Input: likelihood function $f(x) = p(y|x)$ and locations $\{x_i\}_{i=1}^N$, residual precision ϵ, residual threshold τ.
Initial state-likelihood pairs: $\mathcal{D}^{(1)} = \{(x_i, r_i)\}_{i=1}^N$, where $r_i = p(y_i|x_i)$.
$k = 1$.
repeat
  Calculate log-likelihood $\ell_i = \log r_i, \forall i$.
  Find highest point: $m = \text{argmax}_i \, \ell_i$.
  Fit a quadratic lower-bound $h^{(k)}$ according to (32).
  Calculate scaled Gaussian: $f^{(k)}(x) = \exp(h^{(k)}(x))$.
  Calculate residuals and bound: $\hat{r}_i = \max(r_i - f^{(k)}(x_i), \epsilon), \forall i$.
  Set points as residuals for next iteration: $\mathcal{D}^{(k+1)} = \{(x_i, \hat{r}_i)\}_{i=1}^N$.
  $k \leftarrow k + 1$
until $\max_i(\hat{r}_i) \leq \tau$
Output: approximate likelihood $\hat{f}(x) = \sum_k f^{(k)}(x)$.

Gaussian. The precision matrix W_k is found by minimizing the square-error with $\mathcal{D}$ in the log-likelihood space, with the constraint that the log-Gaussian $h^{(k)}(x)$ is a lower bound of the points in $\mathcal{D}$,

$$W_k^* = \underset{W_k}{\text{argmin}} \ \frac{1}{2}\sum_{i=1}^{N}(\ell_i - h^{(k)}(x_i))^2 \qquad (32)$$
$$\text{s.t.} \ \ \ell_i - h^{(k)}(x_i) \geq 0, \forall i,$$

where $\ell_i = \log r_i$. When $W_k = \text{diag}(w_k)$ is a diagonal matrix, then (32) is a quadratic program (see Appendix E). The constraints in (32) ensure that $f^{(k)}$ is a lower bound of $\mathcal{D}^{(k)}$. Finally, the residual points are calculated $\hat{r}_i = r_i - f^{(k)}(x_i), \forall i$ and the next iteration is run on the residual data $\mathcal{D}^{(k+1)} = \{(x_i, \hat{r}_i)\}_{i=1}^N$. After sufficient iterations to reduce all residuals to under a threshold, the approximate likelihood is $\hat{f}(x) = \sum_k f^{(k)}(x)$. Because each iteration forms a lower bound to the residuals, $\hat{f}(x)$ is a lower-bound of the original data $\mathcal{D}$. Algorithm 1 summarizes the procedure, and an example is shown in Fig. 4.

4.2 Reducing KDE mixtures

Kernel density estimation (KDE) is an effective non-parametric method for density estimation [4], and has seen many applications [16, 61, 62]. Formally, given a dataset $D = \{x_1, \ldots, x_N\}$, the KDE using a Gaussian kernel is

$$p(x|D) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{N}(x|x_i, \sigma^2 I), \qquad (33)$$

where σ^2 is the kernel bandwidth. A KDE mixture (MixKDE) was proposed in [16], where each mixture component is a separate KDE over different hierarchical levels of the data. For example, [16] creates a hierarchy of spatial check-in data, from the individual-level, to the district-level, and to the city-level. The spatial distribution for an individual is obtained by mixing the levels,

$$p(x|D) = \sum_{l=1}^{L}\alpha_c p_c(x|D_l), \qquad (34)$$

where D is the complete dataset, $D_1 \subset D_2 \cdots \subset D_L$ are the hierarchical levels (individual to population), $p_c(x|D_c)$ is the KDE for level c, and $\{\alpha_c\}$ are the mixing weights.

One disadvantage of KDE/MixKDE is it requires storing and processing all samples of the dataset in order to compute a likelihood, which may not be possible on devices with limited memory or computation. [16] proposes an efficient approximation using k-d tree, where the KDE at a test point is approximated with a subset of the dataset, defined by the point's k-d tree cells. This approach reduces the computation for calculating the likelihood of the test point, but also suffers inaccuracies when the test point falls close to a cell boundary.

The KDE/MixKDE with Gaussian kernel can be interpreted as a GMM with a very large number of components. Hence, we use DPHEM to reduce the MixKDE in (34) into an equivalent GMM with fewer components (see Fig. 5). As the reduced GMM has fewer components, less memory is required to store the parameters, and computing the test likelihood is more efficient.

5 Experiments

In this section, to show the applicability of DPHEM, we present five experiments: 1) reducing synthetic data of 2D GMMs; 2) using KDE mixtures to model user location data; 3) visual tracking with Bayesian filtering and GMM posteriors; 4) vehicle self-localization with Bayesian filtering; 5) synthetic experiment on belief propagation. We compare with three other density simplification methods, the original HEM [24], VKL [21] and L2U [25]. Experiments were implemented with Matlab on a desktop PC.

5.1 Synthetic data

In this experiment, we reduce the number of components of 2D synthetic GMM data by applying the proposed method (DPHEM) and the other three (HEM, VKL, L2U). We sampled 2,500 points as the means of the synthetic base GMM from a randomly generated KDE with 50 kernels. The component weight is inversely proportional to the distance between the sample point to related kernel center. The same isotropic covariance is used for all components to ensure enough overlap between components. The four simplifying methods are applied to the base GMM to reduce the mixture one component at a time, until only one component remains. For each target K_r, we use a weighted k-means initialization (as in [25]), where the samples are the base component centers and the corresponding weights are the component weights. The best reduced mixture model from 10 random initializations of k-means is selected using a method's corresponding objective criteria (i.e. variational expected log-likelihood for HEM and DPHEM, variational KL for VKL, approximate L_2 for L2U). The similarity between the base and reduced mixtures is then evaluated using KL-divergence (KLD), which is calculated using Monte Carlo approximation with 100,000 samples. The experiment was repeated using 100 base GMMs, and the average KLD and processing time for each K_r is calculated.

Fig. 6a plots the KLD versus the number of reduced components $K_r \in [1, 50]$. Since there are roughly 50 clusters in the base mixture model, L2U and DPHEM estimate reduced GMMs that are equivalent to the base GMM (KLD is near 0) when $K_r > 50$. As K_r decreases, DPHEM maintains a lower KLD than L2U. Although DPHEM and VKL have similar curves when $K_r < 38$, DPHEM has consistently

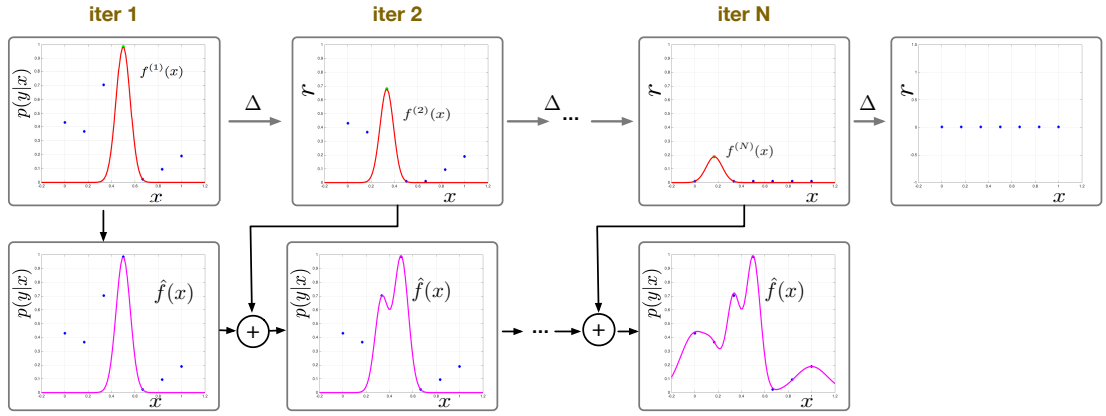


Fig. 4. A 1D example of likelihood approximation using sum of scaled Gaussians. Δ indicates calculation of the new residuals, $\hat{r}_i = r_i - f^{(k)}(x_i)$.

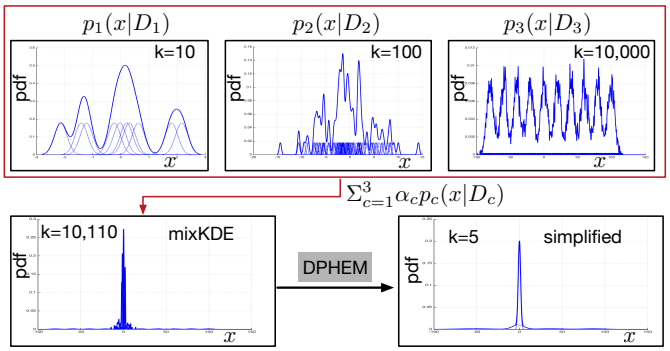


Fig. 5. A 1D example of simplifying a KDE mixture for human location modelling. The top row shows KDEs representing spatial distributions of check-in locations (x-axis represents location) at different hierarchical levels: 10 check-ins of an individual (p_1); 100 check-ins in a local region (p_2); 10,000 check-ins for the whole population (p_3). In the bottom row, an individual's spatial distribution is a KDE mixture, which is a weighted combination of the KDEs, and is simplified using DPHEM. k is the number of components in each KDE.

lower KLD than VKL. On the other hand, the reduced model from HEM has a large KLD, even when a large number of reduced components are used, which indicates that HEM cannot well preserve the density structure. Fig. 6b plots the processing time to reduce a base mixture model to a given number K_r. DPHEM has the lowest computing time – in particular, VKL is slower to converge because it uses two sets of variational parameters.

Fig. 2 shows a typical comparison between DPHEM and the other methods. When reducing to a same number of components, DPHEM preserves the base component better while HEM promotes some small overlapping components because of its multiple instance learning property (Fig. 2a). Comparing to L2U, which tends to preserve more large components, DPHEM also model the small components (Fig. 2c). The performance of DPHEM is mostly similar to VKL, but DPHEM fits better to the base mixture, as VKL approximates two small neighboring base components as one reduced component with large covariance (Fig. 2b).

Finally, we compare DPHEM using different number of virtual samples $N \in \{1, 200, 400, \ldots, 10^5\}$. On this simple dataset, similar results were obtained for values $N > 1$, yielding the same DPHEM curve in Fig. 6. Using $N = 1$ yielded lower KLD when the number of reduced mixture components K_r is much less than the actual number of components in the data. For $N = 1$, processing time was higher due to more iterations caused by slower convergence. On this dataset, the hard-clustering version of DPHEM ($N \to \infty$), which is equivalent to [20], also yields similar results to soft-clustering DPHEM.

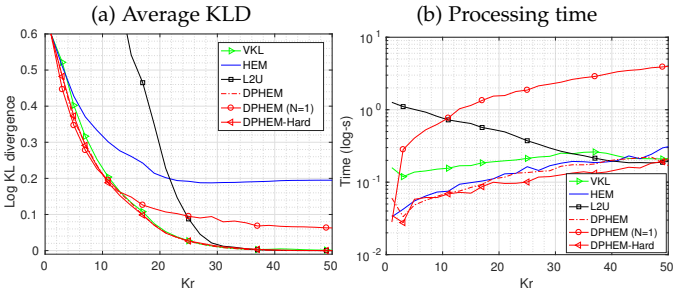


Fig. 6. Experiments on reducing 2D GMMs with 2500 components. (a) Average KL-divergence between the original and reduced models for different number of reduced components K_r. (b) Processing time vs number of reduced components.

5.2 Human location modelling

In this experiment, we apply DPHEM to KDE mixtures (MixKDE) to obtain an efficient and accurate reduced representation. The goal is to model the spatial location of an individual's check-in events from a social media website. Following [16], we use the Gowalla dataset [63] and extract a subset of events occurring in Southern California on weekdays from January to October 2010. The data from January through June are used for training, while the data from July through October are used for testing. The training set contains 64,368 events from 4,281 individuals. For each individual, a MixKDE (Eq. 34) was trained with 3 levels following [16]: the first is individual-level, trained with all the individual's events; the last is population-level; the middle-level is a local region, where we partition the Southern California area into a 9×9 grid of local regions, and train with the one containing the majority of the individual's events (see Fig. 9a). An adaptive bandwidth [16] is selected for each KDE component, which is the distance between current event and its k-th nearest neighbor. A separate set of 25,000 events from the training set are used to obtain the mixing weights α_c. As the original MixKDEs (base model) are computationally inefficient for calculating the likelihood of a new test event, 4 simplifying methods are applied on the base model to obtain reduced GMM representations. The weighted k-means initialization from Section 5.1 is used. We also compare with the k-d tree approximation from [16].

The models are evaluated by randomly selecting 1,000 individuals from the testing data set, and calculating the log-likelihood (LL) of an individual's test events on the individual's MixKDE. Fig. 7 shows a scatter plot comparing

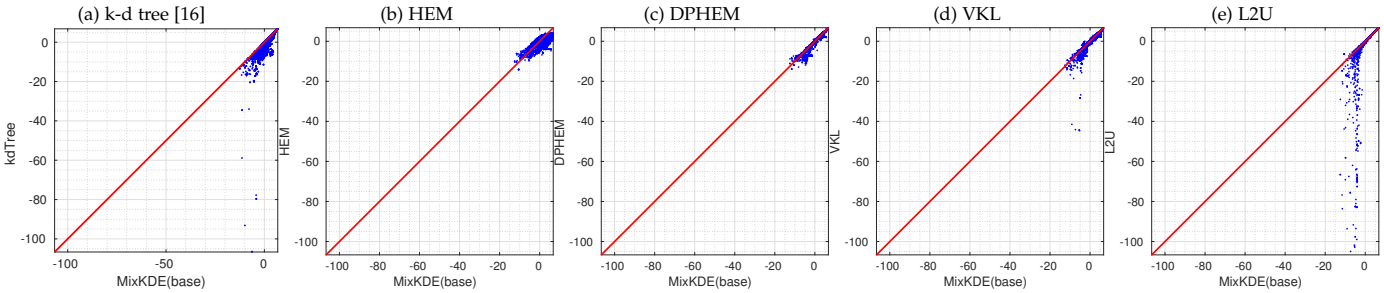


Fig. 7. Scatter plots of the log-likelihoods of test events for approximate models with $K_r = 90$ versus the original KDE mixture.

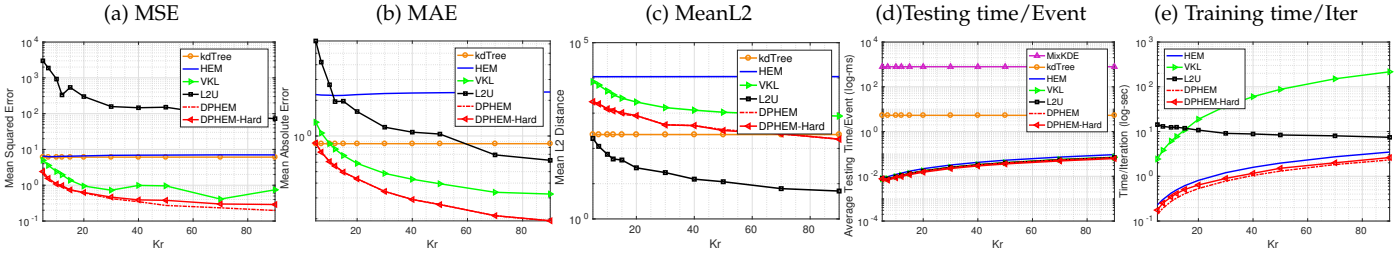


Fig. 8. KDE mixture reduction: (a-b) MSE and MAE of test-event log-likelihoods and (c) mean L2 distance of test-event likelihoods between mixture KDE and approximate models; (d) Average computing time per event; (e) Average learning time per iteration for each method. K_r is the number of components in the approximate models.

the event log-likelihoods under the approximate models ($K_r = 90$) and the original base model (MixKDE). The DPHEM scatter plot is more compact and closer to the diagonal, indicating that DPHEM has log-likelihoods that are the most similar to the original MixKDE. In contrast, HEM, VKL, L2U and k-d tree do not well-preserve the base model's density structure. This is further confirmed by examining the mean-squared error (MSE) and mean absolute error (MAE) between the test event LLs predicted by the original MixKDE and the approximate models in Fig. 8a and Fig. 8b, where DPHEM has the lowest values among the methods. Since low-density regions may tend to dominate the evaluation metrics using LLs, we also show the average L2 distance (MeanL2) between test event *likelihoods* in Fig. 8c. Comparing soft- and hard-clustering methods, as K_r increases the MSE of DPHEM-hard [20] becomes worse than DPHEM, which suggests that the hard-clustering approach loses efficacy when there are a large number of overlapping components.

Comparing the processing time for a test event, the mixture-model approximations (HEM, VKL, DPHEM, L2U) require much less time than kd-tree, e.g., for $K_r = 90$, 0.068 ms vs. 5.302ms. More test time comparisons are shown in Fig. 8b. DPHEM and HEM have the lowest training time per iteration (see Fig. 8c).

Fig. 9b shows an example of MixKDE and approximate mixtures for an individual's spatial distribution. Although all approximate models except for HEM can preserve the main structure of MixKDE, differences in the density maps still exist. For example, the components around location (-117.8, 33.7) are different for each approximate model: some of them are missing in the kdTree approximation; components with small weight are promoted to have high probability in HEM; some small components are missing in VKL approximation; most heavy tails are missing in L2U. Since the model differences may be hard to illustrate because of color scale in density map, the average log-likelihood (meanLL) of the testing events (red dots) on each model are shown, as well as the MSE, MAE and MeanL2. On this example, DPHEM achieves the closest meanLL to MixKDE and lowest MSE, MAE and MeanL2.

5.3 Visual tracking

In this experiment, we apply DPHEM to visual tracking using recursive Bayesian filtering. For visual tracking, y_t is the video frame and x_t is the location of the target. The observation model $p(y_t|x_t)$ is the likelihood (score) that the target is located at x_t in image y_t. Here we use the observation model from compressive tracking (CT) [64], which uses a sparse measurement matrix to extract features for tracking. The likelihood function for a given y_t is approximated as an SSG with isotropic covariances using Algorithm 1 – the likelihood-state pairs are obtained by densely sampling locations and computing the likelihood scores. A simple Gaussian motion model is used, which is based on the estimated velocity from the previous two frames, $p(x_t|x_{t-1}) = \mathcal{N}(x_t|x_{t-1} + \hat{v}_{t-1}, \Sigma)$ where $\Sigma = \sigma^2 I$ is an isotropic covariance matrix (we set $\sigma^2 = 20$) and the estimated velocity is $\hat{v}_{t-1} = \hat{x}_{t-1} - \hat{x}_{t-2}$, with $\hat{x}_t$ as the predicted position at time t. The updated posterior is obtained using (29), which has on average 500 components. DPHEM is then used to reduce the GMM posterior to $K_r = 50$ components. The 50 components with largest weights are used for initialization. Finally, the tracked position is obtained using MAP, $\hat{x}_t = \text{argmax}_{x_t} p(x_t|y_{1:t})$, and the image patch around $\hat{x}_t$ is used to update the likelihood model for the next frame. Fig. 10a illustrates the tracking procedure.

We test our tracking method on a commonly-used benchmark [65] which contains 50 sequences. Since the size of the bounding box in our method is fixed, the evaluation is based on the precision plot of OPE (One Pass Evaluation) proposed in [65], which is the percentage of successfully tracked frames whose center location error is within a certain threshold. We compare our DPHEM tracker with the following baseline tracking inference algorithms (all use the same observation model):

- Dense - dense sampling of candidate locations x_t. The tracked location is the one with maximum score $p(y_t|x_t)$. This was used in the CT tracker [64].

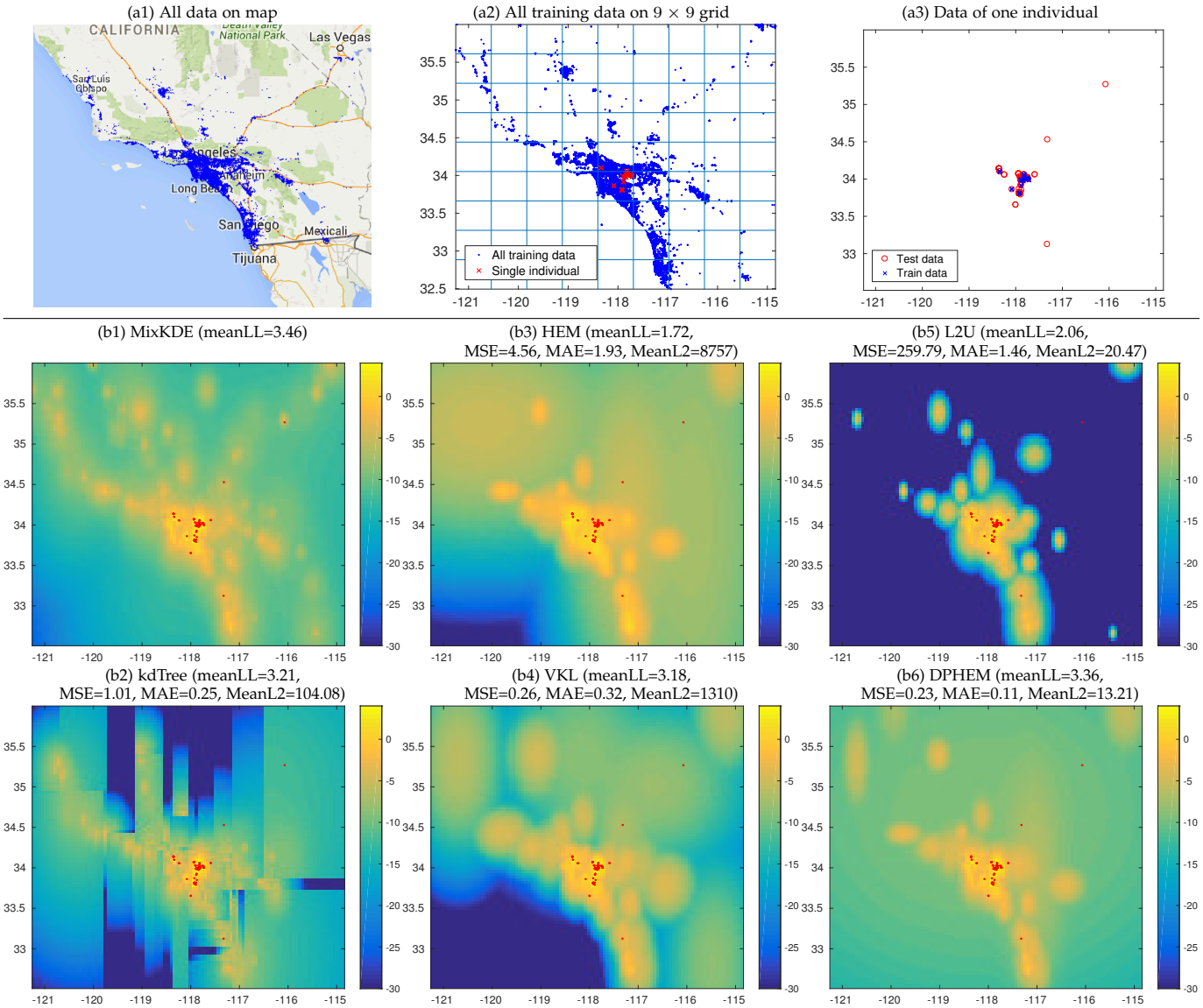


Fig. 9. Example of human location modeling: (a1) All check-in records used for training and testing on map; (a2) All training and testing of one selected individual on a 9×9 grid; (a3) The individual's training and testing data. (b) Log-likelihood map of the individual's spatial distribution estimated from check-in records with mixture KDE model (MixKDE) and the simplified models with $K_r = 90$. Red dots are the test events of this individual. meanLL is the average log-likelihood of the test events. MSE and MAE are the mean squared error and mean absolute error between the test event log-likelihoods predicted by the original MixKDE and the approximate model, while MeanL2 is the mean L2 distance between test event likelihoods. For better visualization, log-likelihood values are truncated at -30.

- OPF - original particle filter where resampling is used to propagate particles to the next frame. The tracked position is the particle with maximum weight.
- IPF - an improved particle filter where only the particle with maximum weight is propagated to the next frame.

For each tracker, we test two versions with and without velocity in the motion model (+Motion, +NoMotion). Since DPHEM + Motion outperforms all other trackers, we also test and show the results by replacing DPHEM with other simplification methods, HEM, VKL and L2U.

Fig. 12a presents the precision plots for the various tracking methods. Our DPHEM+Motion has higher precision than the other inference methods (P@20 of 0.430 for DPHEM vs. 0.418 for IPF+NoMotion). Comparing simplification methods within the GMM tracking framework, DPHEM outperforms L2U, VKL, and HEM. DPHEM significantly outperforms the original particle filter with resampling (OPF), which suggests that using GMMs to represent posteriors has significant advantages over using weighted particles. IPF has better precision than Dense, mainly because the Gaussian diffusion model reduces the chance of selecting outlier points far from the predicted position as the tracked position. However, both methods perform slightly worse when using velocity in the motion model (+Motion).

5.4 Vehicle self-localization

We apply DPHEM to recursive Bayesian filtering for vehicle self-localization. [21] proposed a probabilistic model for vehicle self-localization using roof-mounted cameras and a map of the driving environment. The map is represented as a graph (see Fig. 11a), and probabilistic models describe how the vehicle can traverse the graph. Recursive Bayesian filtering is used to infer the posterior probability of the vehicle's states (location and orientation) on the map, given the visual odometry (observed displacement and orientation change) calculated from the cameras. Fig. 11 shows the inference results for the vehicle's state on the map.

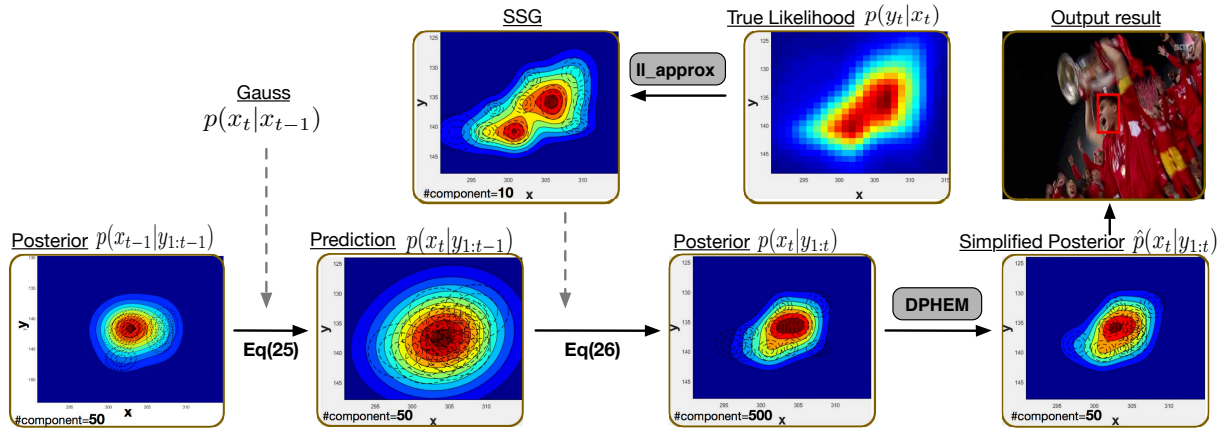


Fig. 10. Pipeline for visual tracking using GMM posteriors and density simplification.

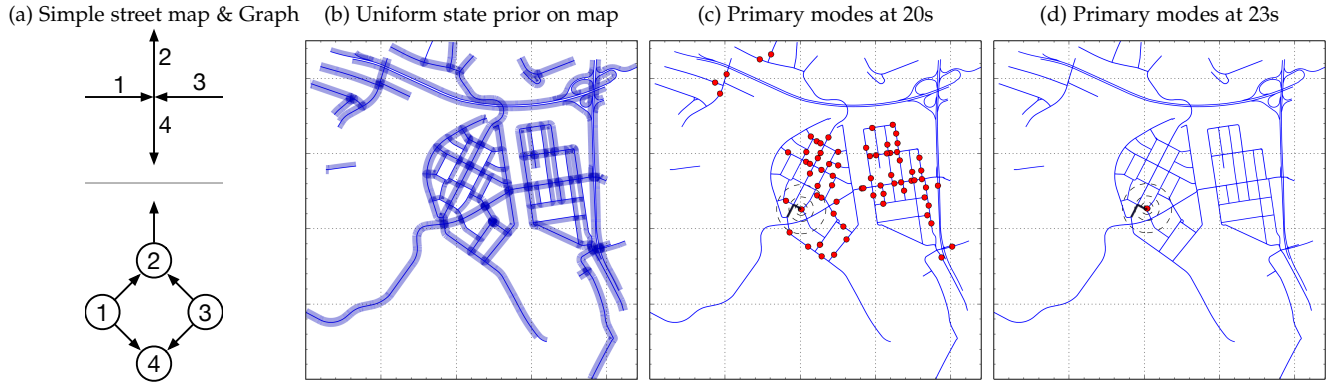


Fig. 11. Vehicle self-localization with GMMs: (a-top) Simple street map of 4 road segments at a crossing, (a-bottom) corresponding graph representation, where each node represents a road segment and a directed edge indicates road connectivity. (b) The prior state distribution is a uniform distribution over all roads on the map; (c) Primary modes of the posterior at 20s – ambiguity of the vehicle's states decreases since the posterior on most road segments becoming extremely small. (d) Primary modes at 23s: a single primary mode remains. The vehicle is considered as localized when there is a single primary mode for 10 contiguous frames.

In [21], the state posterior is represented as a 4D GMM. Street transition and state transition models are applied to each Gaussian mode (component) of the posterior distribution, obtaining a set of predictions for the next frame. Usually one mode leads to one predictive distribution on its successor road segment, but the number of components will increase when there are intersections (multiple transition options). Using the visual odometry observation model, the predictive distributions are adjusted to form the new posterior. For the initialization, the initial state is a uniform distribution on all road segments on the map, which is formed by creating a GMM with 100 evenly-spaced components per kilometer. In order to speed up the calculations, [21] applied VKL to remove components with small weights in the posterior at each time step. In this paper, we apply DPHEM to reduce the complexity.

We use the code released by the author, and follow the same experiment setup and use the same visual odometry dataset as [21]. We ignore video sequences 04 (short highway road) and 06 (symmetric road) which cannot be localized by this probabilistic model, and use stereo odometry for the observation model. In order to examine the influence of different simplifying algorithms on inference for the vehicle's localization, we reduce the posterior distribution at every frame to a given number of components, which is based on the length of the road segment. The average filtering time per frame on the 9 test videos is shown in Fig. 12b. Without using simplification, inference takes on average 8.75s per frame. Although the posterior grows more complex with each new frame, the transition model can help to ignore some roads that are infeasible for the vehicle to move to from the current location. Applying simplification to the posterior can save more than half the computing time, and DPHEM is faster than the other simplification methods. The largest difference in speed appears at $K_r = 30/\mathrm{km}$, which we use for evaluation hereafter.

We next evaluate the localization error. Here we consider the vehicle as localized when there is a single primary mode in 10 contiguous frames, i.e., there exists only one peak in the posterior[3] In each frame, the primary modes are the states in the current posterior with large likelihood, and are found by applying non-maximum suppression on the Gaussian components of the posterior (with a threshold of 30m), and then searching the regions around the remaining components for a local maximum in the posterior.

The localization error with respect to the ground truth is presented in Table 2. Although the final localization error is influenced by many factors in this complex system, with all other parameters kept the same, we obtain similar and sometimes better localization results with DPHEM, as

3. In contrast, [21] defines localization as having a single primary mode in 10 frames *and* this mode is within 20m of the ground-truth. Here we use an evaluation criteria based on a more realistic scenario where the primary modes are used to propose possible locations while driving, without knowledge of the ground-truth.

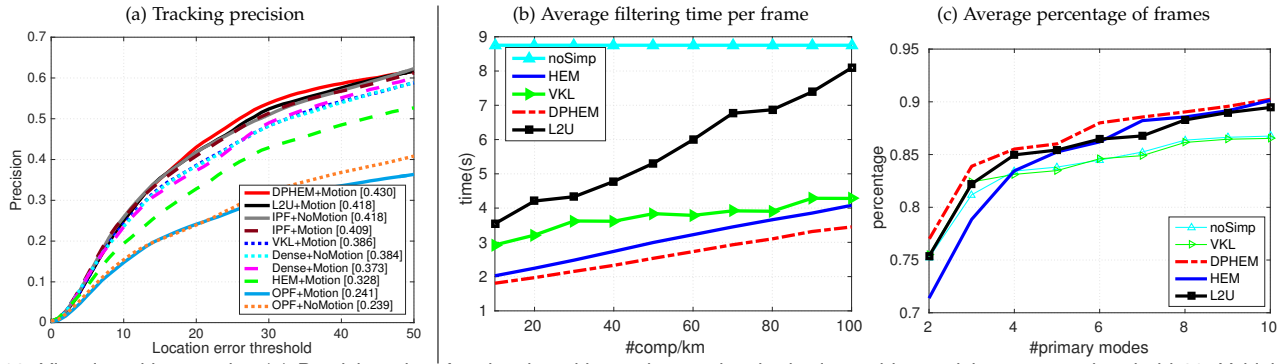


Fig. 12. Visual tracking results: (a) Precision plots for visual tracking – the number in the legend is precision at error threshold 20. Vehicle self-localization results: (b) Average filtering time per frame of 9 video sequences; (c) Average percentage of frames with the number of primary modes less than a given value.

TABLE 2
Average position and angle error of the first 10 localized frames for each test video sequence using stereo visual odometry. "Avg" is the average error over all sequences. Oracle is the projection error of the GPS track onto the map, which represents the best localization result for the map.

Error Type	Methods	Video Sequence 00	01	02	03	05	07	08	09	10	Avg
position (m)	no simplification	1.01	5.99	2.39	3.15	2.66	1.92	121.46	3.98	2.73	16.14
	VKL	0.98	12.30	2.31	3.26	2.86	1.98	136.99	3.66	2.45	18.53
	DPHEM	1.07	15.24	2.15	3.63	2.34	1.57	27.55	3.78	2.66	6.66
	HEM	0.85	10.77	2.94	483.73	2.52	3.75	1.96	2.68	5.23	57.16
	L2U	0.93	2301.95	2.56	471.72	2.37	1.60	0.82	3.11	3.05	309.79
	Oracle	0.75	1.30	1.03	2.53	1.31	0.60	1.08	1.24	1.03	1.21
angle (degrees)	no simplification	0.50	2.81	1.44	0.57	1.91	1.25	1.35	1.28	0.86	1.33
	VKL	0.63	6.72	1.51	0.71	2.21	1.41	0.94	1.01	0.75	1.77
	DPHEM	1.88	0.80	0.83	2.08	1.55	1.77	0.93	1.03	0.91	1.31
	HEM	0.71	4.77	1.39	82.77	0.99	2.54	3.98	1.51	1.20	11.10
	L2U	1.35	40.84	0.70	91.74	2.41	1.81	1.12	1.14	0.67	15.75

compared to the other simplification methods.

Finally, we examine the algorithms before localization occurs when there are multiple primary modes. When a few primary modes exist, the system could be used to give good suggestions to the driver. Fig. 12c plots the percentage of frames with less than a given number of primary modes. DPHEM has higher percentage of frames having less primary modes, which means the ambiguity decreases faster than other methods, and good suggestions about the vehicle's location can be proposed at earlier times. Even though the number of primary modes decreases faster, the localization error shown in Table 2 shows that the accuracy does not diminish. Note that L2U and HEM also have a high percentage of frames with less primary modes, but exhibit large localization errors in Table 2. For these methods, although the ambiguity has been reduced quickly and eventually a single mode remains, the final mode is not close to the ground-truth location.

5.5 Belief propagation

In this section, we apply DPHEM to belief propagation (BP), which is a useful tool for inference in graphical models that is based on a sequence of local message passing. In a single-connected graph, BP can perform exact inference, but returns an approximate result when loops exist. Normally messages are interpreted as vectors for discrete variables, and sufficient statistics such as mean and variance for Gaussian variables. However, there is usually no tractable analytic representation for messages in BP when continuous and non-Gaussian variables exist in the graphical model [36]. [36] proposed non-parametric BP, where messages are represented with a Gaussian KDE. Each message update involves a products of GMMs, which results in a large number of components. [36] proposed to use Gibbs sampling to approximate the message products as a GMM with fewer components.

In this experiment, we use a small undirected graph with 4 nodes (see Fig. 13a), in order to be able to compute exact message passing as a baseline for comparison. The self-potentials are assumed to be a GMM with 2 components, with means sampled uniformly over $[-4, 4]$, covariance 1 and random component weights. Each pairwise edge-potential is assumed to be a single Gaussian with 0 mean and inverse variance given by the corresponding element ϕ_{ij} of $\phi = \begin{bmatrix} 1 & 0.2 & 0.4 & 0.6 \\ 0.2 & 1 & 0.01 & 0 \\ 0.4 & 0.01 & 1 & 0.8 \\ 0.6 & 0 & 0.8 & 1 \end{bmatrix}$ We compare the following methods for BP:

- Exact: messages are exact product of mixtures;
- Gibbs: messages are modeled with Gaussian kernel non-parametric density estimation, and message products are found by local Gibbs sampling [36] (we use 1,024 samples);
- DPHEM, HEM, VKL, L2U: messages are the exact product of mixtures, which are simplified whenever the number of components exceeds K_r. The weighted k-means initialization is used.

We compare the various methods by calculating the KLD between the marginal densities when using the approximate messages and those when using the exact messages (no simplification). The results are averaged over 100 trials.

We focus on the first 3 iterations of BP, since computing the exact messages beyond 3 iterations becomes intractable. Fig. 13(b-d) shows the error (average KLD over the 4 nodes) of the various methods in each BP iteration. In this synthetic experiment, the number of components grows very fast and most of them are overlapped with each other. With good initialization, L2U and DPHEM can preserve the marginals

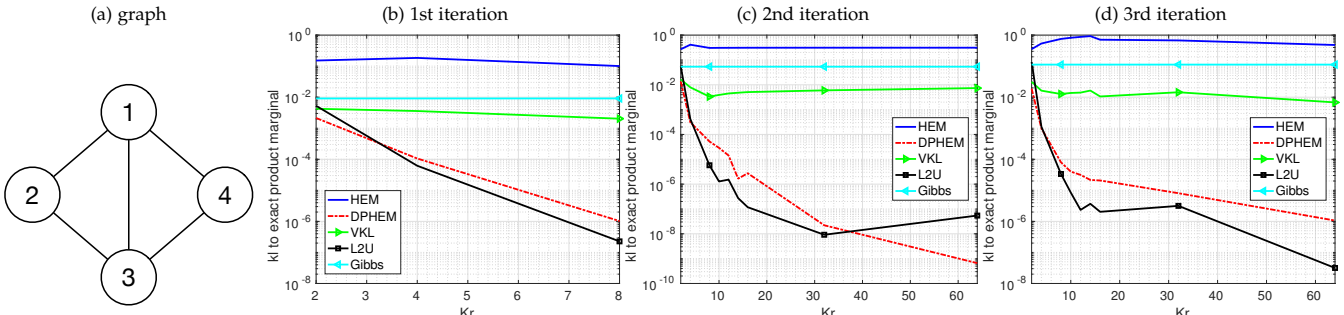


Fig. 13. Belief propagation experiment: (a) the graph; (b-d) average KL divergence between different approximate marginals and the exact product marginal, at each iteration of belief propagation.

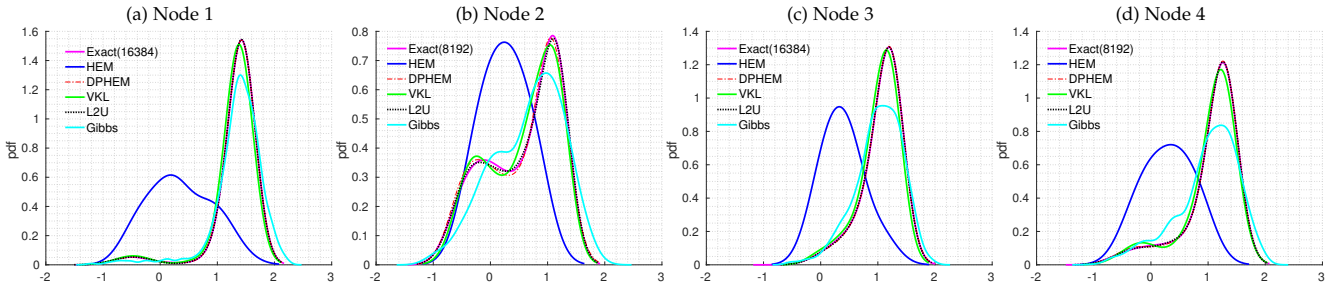


Fig. 14. The marginal density (belief) at each node after the 3rd iteration of belief propagation. The number of components in the exact marginal density at each node is shown in the legend. Gibbs sampling uses 1024 samples, while simplification methods use $K_r = 4$.

well with most KLD less than 10^{-4}. VKL and Gibbs also can preserve the marginal, but with higher KLD around 10^{-2}. In contrast, HEM does not preserve the marginal (KLD of 0.1). Fig. 14 shows an example of marginal distributions after the 3rd iteration. DPHEM and L2U overlap the exact posterior. VKL also does well when modeling the largest mode, but distorts the lower probability modes (e.g., in nodes 2 and 4).

6 Conclusion

In this paper, we proposed a method for simplifying probabilistic mixture models, and in particular, Gaussian mixture models. Our method is based on a variational lower bound of the expected log-likelihood of a set of virtual samples, drawn from the input mixture model. We also proposed an algorithm for approximating a likelihood function as a sum of scaled Gaussians, which was used for recursive Bayesian filtering with GMM posteriors. Finally, we demonstrated the efficacy of our algorithm in recursive Bayesian filtering and KDE reduction, where mixture models with large number of components are common, with specific applications of visual tracking, vehicle self-localization, and human location modelling. We also demonstrate the applicability of our algorithm to belief propagation with GMM messages. In future work, we will explore the feasibility of our inference algorithm on higher dimensional distributions, such as those in Gaussian process regression, so that it can be applied to more computer vision tasks.

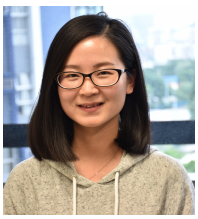
Acknowledgments

The authors would like to thank: A Ihler and M Mandel for KDE toolbox; E Cho, SA Myers and J Leskovec for the Gowalla dataset in [63]; Y Wu, J Lim and MH Yang for the tracking dataset and codes in [66]; K Zhang, L Zhang and MH Yang for the codes in [64]; MA Brubaker, A Geiger and R Urtasun for the dataset and codes for self-localization in [21]. This work was supported by the Research Grants Council of the Hong Kong SAR, China (CityU 110513).

References

[1] D. M. Titterington, *Statistical analysis of finite mixture distributions*. John Wiley and Sons, Inc., New York, NY., 1985.
[2] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.
[3] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.
[4] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.
[5] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *ECCV*, 2002.
[6] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *IJCV*, vol. 77(1-3), pp. 125–41, 2008.
[7] F. Bardet, T. Chateau, and D. Ramadasan, "Illumination aware mcmc particle filter for long-term outdoor multi-object simultaneous tracking and classification," in *ICCV*, 2009, pp. 1623–1630.
[8] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *CVPR*, 2012.
[9] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *CVPR*, 2012.
[10] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via multi-task sparse learning," in *CVPR*, 2012.
[11] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *CVPR*, 2013.
[12] D. Wang and H. Lu, "Visual tracking via probability continuous outlier model," in *CVPR*, 2014.
[13] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2356–2368, 2014.
[14] D. Varas and F. Marques, "Region-based particle filter for video object segmentation," in *CVPR*, 2014.
[15] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *IJCV*, vol. 111(2), pp. 213–28, 2015.
[16] M. Lichman and P. Smyth, "Modeling human location data with mixtures of kernel densities," in *KDD*, 2014, pp. 35–44.
[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.
[18] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE TPAMI*, vol. 24(3), pp. 381–96, 2002.
[19] S. J. Gershman and D. M. Blei, "A tutorial on bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, 2012.
[20] J. Goldberger and S. T. Roweis, "Hierarchical clustering of a mixture model," in *NIPS*, 2004, pp. 505–512.
[21] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization," *IEEE TPAMI*, vol. 38(4), no. 4, pp. 652–65, 2016.
[22] J. Dhillon, "Differential entropic clustering of multivariate gaussians," *NIPS*, vol. 19, p. 337, 2007.

[23] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *JMLR*, vol. 6, pp. 1705–49, 2005.
[24] N. Vasconcelos and A. Lippman, "Learning mixture hierarchies," in *NIPS*, 1998, pp. 606–612.
[25] K. Zhang and J. T. Kwok, "Simplifying mixture models through function approximation," *IEEE TNN*, vol. 21(4), pp. 644–58, 2010.
[26] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE TPAMI*, vol. 29(3), pp. 394–410, 2007.
[27] T.-J. Cham and J. M. Rehg, "A multiple hypothesis approach to figure tracking," in *CVPR*, 1999.
[28] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE TSP*, vol. 50(2), pp. 174–88, 2002.
[29] M. L. Psiaki, J. R. Schoenberg, and I. T. Miller, "Gaussian sum reapproximation for use in a nonlinear filter," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 2, pp. 292–303, 2015.
[30] M. L. Psiaki, "Gaussian mixture nonlinear filtering with resampling for mixand narrowing," *IEEE TSP*, vol. 64(21), pp. 5499–512, 2016.
[31] J. M. Coughlan and S. J. Ferreira, "Finding deformable shapes using loopy belief propagation," in *ECCV*, 2002, pp. 453–468.
[32] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005.
[33] A. T. Ihler and D. A. McAllester, "Particle belief propagation," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 256–263.
[34] M. Isard, J. MacCormick, and K. Achan, "Continuously-adaptive discretization for message-passing algorithms," in *NIPS*, 2009, pp. 737–44.
[35] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *IEEE TSP*, vol. 58(1), pp. 269–280, 2010.
[36] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," *Communications of the ACM*, vol. 53, no. 10, pp. 95–103, 2010.
[37] L. Sigal, M. Isard, H. Haussecker, and M. J. Black, "Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation," *IJCV*, vol. 98(1), pp. 15–48, 2012.
[38] Y. Ma, J. Zhu, and D. Baron, "Approximate message passing algorithm with universal denoising and gaussian mixture learning," *IEEE TSP*, vol. 64(21), pp. 5611–22, 2016.
[39] S. Kwak, T. Lim, W. Nam, B. Han, and J. H. Han, "Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering," in *ICCV*, 2011.
[40] O. Muller, M. Ying Yang, and B. Rosenhahn, "Slice sampling particle belief propagation," in *ICCV*, December 2013.
[41] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu, "Spm-bp: Sped-up patchmatch belief propagation for continuous mrfs," in *ICCV*, 2015.
[42] T. Jebara, Y. Song, and K. Thadani, "Spectral clustering and embedding with hidden markov models," in *Machine Learning: ECML 2007*, 2007, pp. 164–175.
[43] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-invariant dynamic texture recognition using a bag of dynamical systems," in *CVPR*, 2009.
[44] P. Bruneau, M. Gelgon, and F. Picarougne, "Parsimonious reduction of gaussian mixture models with a variational-bayes approach," *Pattern Recognition*, vol. 43, no. 3, pp. 850–858, 2010.
[45] A. Mumtaz, E. Coviello, G. R. Lanckriet, and A. B. Chan, "Clustering dynamic textures with the hierarchical EM algorithm for modeling video," *IEEE TPAMI*, vol. 35, no. 7, pp. 1606–1621, 2013.
[46] N. Vasconcelos, "Image indexing with mixture hierarchies," in *CVPR*, 2001.
[47] K. Ellis, E. Coviello, and G. R. Lanckriet, "Semantic annotation and retrieval of music using a bag of systems representation." in *International Society for Music Information Retrieval Conference(ISMIR)*, 2011, pp. 723–728.
[48] E. Coviello, A. Mumtaz, A. B. Chan, and G. R. Lanckriet, "Growing a bag of systems tree for fast and accurate classification," in *CVPR*, 2012.
[49] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*, 1998, pp. 355–368.
[50] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends® in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.
[51] I. Csisz and G. Tusnády, "Information geometry and alternating minimization procedures," *Statistics and decisions, Supplemental Issue 1*, pp. 205–237, 1984.
[52] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999.
[53] T. S. Jaakkola, "10 tutorial on variational approximation methods," *Advanced mean field methods: theory and practice*, p. 129, 2001.
[54] E. Coviello, A. B. Chan, and G. R. Lanckriet, "Clustering hidden markov models with variational hem," *JMLR*, vol. 15(1), pp. 697–747, 2014.
[55] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *ICASSP*, 2007.
[56] S. Hong and B. Han, "Visual tracking by sampling tree-structured graphical models," in *ECCV*, 2014, pp. 1–16.
[57] A. R. Zamir and M. Shah, "Accurate image localization based on google maps street view," in *ECCV*, 2010, pp. 255–268.
[58] G. Vaca-Castano, A. R. Zamir, and M. Shah, "City scale geo-spatial trajectory estimation of a moving camera," in *CVPR*, 2012.
[59] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *ICRA*, 2012, pp. 1643–49.
[60] Z. Chen, "Bayesian filtering: From kalman filters to particle filters, and beyond," *Statistics*, vol. 182, no. 1, pp. 1–69, 2003.
[61] A. Sadilek, H. A. Kautz, and V. Silenzio, "Modeling spread of disease from social interactions." in *ICWSM*, 2012.
[62] S. J. Vaughan-Nichols, "Will mobile computing's future be location, location, location?" *Computer*, vol. 42, no. 2, pp. 14–17, 2009.
[63] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*, 2011, pp. 1082–90.
[64] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *ECCV*, 2012, pp. 864–877.
[65] Y. Wu, J. Lim, and M.-H. Yang, "Online Object Tracking: A Benchmark," in *CVPR*, 2013.
[66] ——, "Object tracking benchmark," *IEEE TPAMI*, vol. 37(9), pp. 1834–48, 2015.



Lei Yu received the B.S. and M.S. degree in Information and Computing Science, and Computer Software and Theory from the Hunan Normal University, Changsha, China, in 2010 and 2013, respectively. She is currently working towards the PhD degree in Computer Science at the City University of Hong Kong. Her research interests include Computer Vision, Machine Learning and Optimization.

Tianyu Yang received the B.E. degree from Liaocheng University, Liaocheng, China, and the M.E. degree from Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, in 2010 and 2013, respectively. He is currently a PhD student at City University of Hong Kong, China. His current research interests include visual tracking and deep learning.

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.