GMTORCH: Computing with Gaussian Mixture Uncertainty in PyTorch

Ali Darijani[□#¶], Benedikt Stratmann[#¶⊥], Genís Floriach Pigem[#¶], Jürgen Beyerer[⊛⊥], Luisa Hoffmann[⊥], and Michael Heizmann[⊥]

***Abstract*— Gaussian mixture models (GMMs) can represent multimodal, non-Gaussian uncertainty while retaining analytical structure. Scientific workflows, however, require more than fitting a mixture to observations: uncertain quantities must remain available for subsequent transformations, arithmetic, conditioning, and information combination. We present GMTORCH, a PyTorch framework for computing with GMM-represented uncertain scalar and vector quantities. Mixtures may be constructed from parameters or estimated from data; every mixture-valued result remains a complete distribution that can be evaluated, sampled, summarized, conditioned, or used in a subsequent calculation. Marginalization, conditioning, affine maps, sums and differences under independence, and normalized density products are evaluated analytically and can be composed with assumptions made explicit by the selected operations and with component counts tracked. When nonlinear maps leave the finite-GMM family, or exact operations produce impractically many components, the framework provides explicit sample–transform–refit propagation and mixture reduction. Full-covariance computations use Cholesky-based linear algebra and blocked pairwise kernels. Reference checks compare exact operations with independently assembled formulas, and a single-epoch localization example illustrates sequential likelihood-based updating. The contribution is a consistent software realization of established GMM calculus, with exact and approximate steps made explicit.**

I. Introduction

Gaussian mixture models (GMMs) approximate complex and multimodal densities [1], [2] and support uncertainty quantification [3]. In computational workflows, however, density estimation is only a starting point: transformations, observations, and arithmetic must act on the represented uncertainty and return distributions usable in later calculations. Their probabilistic assumptions, including independence, must remain explicit.

[⊛]Idea[□]Mathematics[#]Software[¶]Writing[⊥]Critique

[□#¶]Ali Darijani is affiliated with Fraunhofer IOSB and the Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology, Karlsruhe, Germany. ali.darijani@rwth-aachen.de

[#¶⊥]Benedikt Stratmann is affiliated with Fraunhofer IOSB, Karlsruhe, Germany. benedikt.stratmann@iosb.fraunhofer.de

[#¶]Genís Floriach Pigem is affiliated with the Technical University of Munich, Munich, Germany. genis.floriach@gmail.com

[⊛⊥]Jürgen Beyerer is affiliated with Fraunhofer IOSB and the Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology, Karlsruhe, Germany. juergen.beyerer@iosb.fraunhofer.de

[⊥]Luisa Hoffmann is affiliated with the Institute of Industrial Information Technology (IIIT), Karlsruhe Institute of Technology, Karlsruhe, Germany. luisa.hoffmann@kit.edu

[⊥]Michael Heizmann is affiliated with the Institute of Industrial Information Technology (IIIT), Karlsruhe Institute of Technology, Karlsruhe, Germany. michael.heizmann@kit.edu

Marginalization, conditioning, affine maps, independent sums and differences, and normalized density products preserve finite Gaussian mixtures [4]. Existing software supports estimation, conditional modeling, or subsets of this algebra, as reviewed in Section II. Integrating these capabilities in a common representation enables workflows across operations, but also requires explicit approximation when nonlinear maps leave the finite-GMM family or pairwise operations produce impractically many components.

GMTORCH provides this integration through a common PyTorch representation for model-specified, estimated, and calculated mixtures. Results support density evaluation, sampling, moments, and subsequent operations. The contribution is a composable software realization of established GMM calculus, with operation assumptions and component-count effects stated explicitly, rather than new formulas. Analytical operations are complemented by sample–transform–refit surrogates for nonlinear maps, including dimension-changing ones, and sample-based mixture reduction. Reference checks, a nonlinear density comparison, and a single-epoch localization example illustrate the resulting workflows.

II. Gaussian Mixture Representation

A. Representation and Estimation

A finite *Gaussian mixture model* (GMM) represents an uncertain random vector $X \in \mathbb{R}^d$ as

$$p_X(x) = \mathcal{G}_X(x|g_X) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|m_i, \Sigma_i), \qquad \pi_i \geq 0, \qquad \sum_i \pi_i = 1, \tag{1}$$

where π_i are the mixing weights, $m_i \in \mathbb{R}^d$ are the component means, and positive-definite $\Sigma_i \in \mathbb{R}^{d\times d}$ are the component covariances. The collection $g_X = (\pi_i, m_i, \Sigma_i)_{i=1}^K$ contains the mixture parameters. Compared with a single Gaussian, this representation retains multimodality while remaining finite and analytically tractable for several probabilistic operations. Its mean and covariance follow directly from the laws of total expectation and total covariance,

$$\mathbb{E}[X] = \sum_i \pi_i m_i, \qquad \mathbb{V}[X] = \sum_i \pi_i(\Sigma_i + m_i m_i^\top) - \mathbb{E}[X]\mathbb{E}[X]^\top. \tag{2}$$

The parameters g_X may be specified from a physical or geometric model or estimated from observations. GMTORCH

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

provides full-covariance *expectation-maximization* (EM) estimation for the latter case. For observations $x_n \in \mathbb{R}^d$, $n = 1, \ldots, N$, and fixed component count K, the standard unregularized procedure alternates between an E-step, which computes posterior component responsibilities, and an M-step, which updates the mixture parameters [5], [6]. The E-step is [6, Sec. 9.2.2]

$$r_{ni}^{(t)} = \frac{\pi_i^{(t)} \mathcal{N}(x_n | m_i^{(t)}, \Sigma_i^{(t)})}{\sum_{j=1}^{K} \pi_j^{(t)} \mathcal{N}(x_n | m_j^{(t)}, \Sigma_j^{(t)})}. \tag{3}$$

The M-step uses these responsibilities to obtain

$$\begin{aligned} N_i^{(t)} &= \sum_{n=1}^{N} r_{ni}^{(t)}, \qquad \pi_i^{(t+1)} = \frac{N_i^{(t)}}{N}, \\ m_i^{(t+1)} &= \frac{1}{N_i^{(t)}} \sum_{n=1}^{N} r_{ni}^{(t)} x_n, \\ \Sigma_i^{(t+1)} &= \frac{1}{N_i^{(t)}} \sum_{n=1}^{N} r_{ni}^{(t)} d_{ni}^{(t)} d_{ni}^{(t)\top}. \end{aligned} \tag{4}$$

Here t indexes iterations, $r_{ni}^{(t)}$ is the responsibility of component i for observation n, and $d_{ni}^{(t)} = x_n - m_i^{(t+1)}$. The procedure starts from initialized mixture parameters and assumes positive effective counts and well-defined Gaussian densities; a global optimum is not guaranteed [6]. Estimation is deliberately an entry point rather than the endpoint: once constructed, the same mixture represents an uncertain quantity that can be transformed, combined with other uncertain quantities, conditioned on observations, and approximated when closure is lost. These downstream semantics are independent of origin: a mixture specified from a model, estimated from data, or returned by a previous calculation is the same kind of computable uncertain quantity.

The equations give textbook unregularized EM. The implementation adds λI_d to each covariance update (default $\lambda = 10^{-6}$), where I_d is the $d \times d$ identity matrix. It adds $10\epsilon_{\text{mach}}$ to each effective count before normalizing the counts to weights; ϵ_{mach} is the machine epsilon of the selected floating-point tensor type (`dtype`).

Sample-based calculations depend not only on the number of samples but also on their coverage of the represented distribution. Sobol sequences provide a low-discrepancy point set in the unit cube [7]. The component-stratified sampler allocates integer counts by largest-remainder rounding of $N\pi_i$, so realized proportions may differ slightly from weights and small positive-weight components can receive no points. In Fig. 2, Sobol points are used only for display; the nonlinear fit uses i.i.d. pseudorandom samples.

B. Related Software

General-purpose tools already provide mature GMM fitting and density evaluation. These include SCIKIT-LEARN [8], POMEGRANATE [9], [10], and the PYTORCH-based TORCHGMM and PYCAVE packages [11], [12]. Current POMEGRANATE releases likewise represent specified and fitted mixtures as PYTORCH-backed distributions [10]. Other packages emphasize conditional mixture models: GMR and CGMM support substantial parts of conditioning, regression, or conditional sampling [13], [14]. PYROBOLEARN documents GMM fitting, conditioning, affine maps, density products between GMMs, and reduction to a single Gaussian by moment matching [15], [16].

TABLE I
DOCUMENTED GMM-LEVEL CAPABILITIES OF RELATED SOFTWARE.

Software	Cond.	Aff.	Indep. $\pm$	Density prod.	Non-linear	Reduction
SCIKIT-LEARN	–	–	–	–	–	–
POMEGRANATE	–	–	–	–	–	–
TORCHGMM/ PYCAVE	–	–	–	–	–	–
GMR	✓	–	–	–	–	MM*
CGMM	✓	–	–	–	–	–
PYROBO LEARN	✓	✓	–	✓	–	MM*
GMTORCH	✓	✓	✓	✓	≈	≈

✓ documented capability; – no built-in operation found; ≈ approximate; MM* moment matching to a single Gaussian. GMR also provides unscented-transform utilities for individual Gaussians.

Table I therefore does not assert priority for the individual operations. Instead, it locates the contribution of GMTORCH in their integration: construction and estimation feed a common PYTORCH representation that remains available for exact multi-mixture calculations, explicit approximation, and further downstream computation. A dash denotes functionality not found in the examined public documentation, rather than proof that it cannot be implemented with the package.

III. EXACT GMM CALCULUS

The calculus maps one or more GMM-represented uncertain quantities to a new mixture that can be used in the next calculation. The formulas follow from standard Gaussian identities applied componentwise [6] and are summarized as an uncertainty calculus in [4]; the software contribution is a consistent realization with explicit assumptions and output complexity. An exact result is not merely a collection of updated parameters: it remains a normalized distribution with density evaluation, sampling, moments, conditioning, marginalization, and further propagation available. Here, exact denotes a distributional operation without sampling or model-order approximation; numerical evaluation remains subject to floating-point limitations. Component counts refer to the constructed representation. Removing zero-weight components or consolidating identical components preserves the distribution; merging distinct components generally introduces approximation.

A. Marginalization and Affine Maps

For an index set I, the marginal X_I is obtained by selecting the corresponding entries of every mean and covariance: $m_i' = (m_i)_I$ and $\Sigma_i' = (\Sigma_i)_{II}$. The result is exact and retains K components.

For an affine map $Y = AX + b$, each component transforms as

$$\pi_i' = \pi_i, \qquad m_i' = Am_i + b, \qquad \Sigma_i' = A\Sigma_i A^\top. \tag{5}$$

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

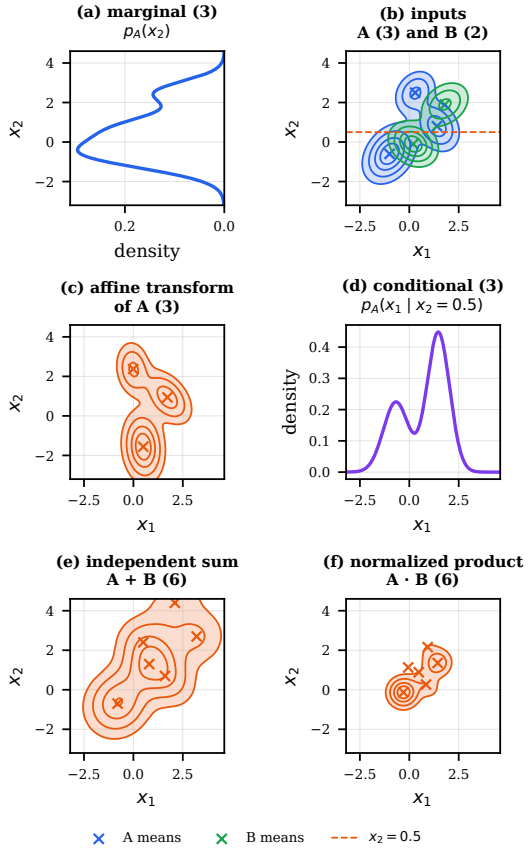


Fig. 1. Exact GMTORCH operations on fixed planar input mixtures A and B. The panels show the sideways marginal $p_A(x_2)$, with the zero of its density axis at the input-facing right edge, the overlaid input densities and component means, with the dashed line marking $x_2 = 0.5$, an affine map of A, the corresponding conditional $p_A(x_1 \mid x_2 = 0.5)$, independent addition, and their normalized density product. Constructed component counts are shown in parentheses; each contour field for a two-dimensional mixture is normalized by that mixture's peak.

This preserves K; the implementation requires A to have full row rank so that the output covariances remain positive definite. Nonzero scalar scaling, translation, and two- and three-dimensional rotations are special cases.

B. Conditioning

Let $Z = [X^\top, Y^\top]^\top$ be jointly represented by a K-component mixture. Partition the parameters of component i as

$$m_i = \begin{bmatrix} m_{X,i} \\ m_{Y,i} \end{bmatrix}, \qquad \Sigma_i = \begin{bmatrix} \Sigma_{XX,i} & \Sigma_{XY,i} \\ \Sigma_{YX,i} & \Sigma_{YY,i} \end{bmatrix}. \tag{6}$$

After observing a nonempty proper subset of coordinates, represented by the subvector $Y = y^*$, the conditional component parameters and weights are

$$\begin{aligned} m_i^* &= m_{X,i} + \Sigma_{XY,i}\Sigma_{YY,i}^{-1}(y^* - m_{Y,i}), \\ \Sigma_i^* &= \Sigma_{XX,i} - \Sigma_{XY,i}\Sigma_{YY,i}^{-1}\Sigma_{YX,i}, \\ \pi_i^* &= \frac{\pi_i \mathcal{N}(y^* | m_{Y,i}, \Sigma_{YY,i})}{\sum_{r=1}^{K} \pi_r \mathcal{N}(y^* | m_{Y,r}, \Sigma_{YY,r})}. \end{aligned} \tag{7}$$

Thus conditioning changes both component parameters and their relative plausibility. It is exact and retains K components.

C. Independent Sums and Differences

Let $X, Y \in \mathbb{R}^d$ be independent, with K_X and K_Y components. Their sum is the exact convolution

$$X + Y \sim \sum_{i=1}^{K_X} \sum_{j=1}^{K_Y} \pi_{X,i}\pi_{Y,j}\mathcal{N}(m_{X,i} + m_{Y,j}, \Sigma_{X,i} + \Sigma_{Y,j}). \tag{8}$$

The difference changes the second mean to $-m_{Y,j}$ and has the same covariance. Thus ordinary arithmetic combines both values and their uncertainties, but the independence assumption is essential: marginal mixtures alone do not determine the sum of correlated quantities. Both operations produce $K_X K_Y$ components.

The current version represents reusable distributions but does not track dependence between mixture objects, including quantities derived from a common source. Callers must establish independence; copying or reusing a mixture does not establish it. Independent subtraction of identical marginals represents independent copies, not the deterministic identity $X - X = 0$.

D. Normalized Density Products

Two densities over the same variable can be combined through their normalized product:

$$p_{\mathrm{f}}(x) = \frac{p_a(x)p_b(x)}{\int p_a(x)p_b(x)\,\mathrm{d}x}. \tag{9}$$

For each component pair (i, j), define

$$\begin{aligned} \Sigma_{ij} &= (\Sigma_{a,i}^{-1} + \Sigma_{b,j}^{-1})^{-1}, \\ m_{ij} &= \Sigma_{ij}(\Sigma_{a,i}^{-1}m_{a,i} + \Sigma_{b,j}^{-1}m_{b,j}), \\ c_{ij} &= \mathcal{N}(m_{a,i} | m_{b,j}, \Sigma_{a,i} + \Sigma_{b,j}), \\ \pi_{ij}^* &= \frac{\pi_{a,i}\pi_{b,j}c_{ij}}{\sum_{r=1}^{K_a}\sum_{s=1}^{K_b}\pi_{a,r}\pi_{b,s}c_{rs}}. \end{aligned} \tag{10}$$

The overlap coefficient c_{ij} accounts for the compatibility of the two components. The operation is exact and produces $K_a K_b$ components. It is a Bayesian update when one factor represents a prior and the other is proportional to a likelihood for the same latent variable; otherwise, it is a normalized density product.

Exact operations may be chained whenever their dimensional and probabilistic assumptions are compatible. The chain remains exact, but pairwise arithmetic and products can multiply the component count at successive steps. Table II summarizes this computational contract.

Figure 1 illustrates exact operations applied to the same fixed input mixtures, not a sequential chain of updates. The panels show marginalization, conditioning, an affine map, independent addition, and a normalized density product.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

TABLE II

CONTRACT OF THE GMTORCH UNCERTAINTY CALCULUS. COMPONENT COUNTS DESCRIBE THE CONSTRUCTED REPRESENTATION BEFORE OPTIONAL PRUNING OR MERGING.

Operation	Meaning	Principal assumption	Exact	Components
Marginalization	retain selected coordinates	joint mixture available	yes	K
Conditioning	update after observing coordinates	finite observation of a nonempty proper coordinate subset	yes	K
Affine map	$Y = AX + b$	full-row-rank A	yes	K
Sum/difference	$Z = X \pm Y$	equal dimension and stochastic independence	yes	$K_X K_Y$
Density product	normalized $p_a(x)p_b(x)$	common variable and base measure	yes	$K_a K_b$
Nonlinear map	fitted surrogate for $f(X)$	sampling and regularized refitting; support need not be preserved	no	chosen K_Y
Reduction	lower-order representation	selected approximation strategy	no	chosen K_{target}

E. Exact Density Comparisons

Unlike the preceding mixture-valued operations, density comparisons return scalar functionals of the represented distributions. Mixtures of equal dimension can be compared without sampling because the integral of a product of two Gaussian densities is available analytically. For mixtures

$$p(x) = \sum_i \pi_i \mathcal{N}(x|m_i, \Sigma_i), \quad q(x) = \sum_j \rho_j \mathcal{N}(x|n_j, \Lambda_j),$$
$$J_{pq} = \sum_{i,j} \pi_i \rho_j \mathcal{N}(m_i|n_j, \Sigma_i + \Lambda_j). \tag{11}$$

the self-overlaps follow by setting the two arguments equal. For densities p and q, define

$$J_p = \int p(x)^2 \, \mathrm{d}x, \qquad J_q = \int q(x)^2 \, \mathrm{d}x,$$
$$J_{pq} = \int p(x) q(x) \, \mathrm{d}x. \tag{12}$$

GMTORCH derives four comparison functionals from these overlap terms [17]–[19]:

$$D^2_{L^2}(p, q) = J_p - 2J_{pq} + J_q, \tag{13}$$

$$D_{\mathrm{CS}}(p, q) = -\log\left(\frac{J^2_{pq}}{J_p J_q}\right), \tag{14}$$

$$\mathrm{JR}_2(p, q) = -\log\left(\frac{J_p + 2J_{pq} + J_q}{4}\right) + \frac{\log J_p + \log J_q}{2}, \tag{15}$$

$$D_{\mathrm{tSL}}(p : q) = \frac{J_p - 2J_{pq} + J_q}{\sqrt{1 + 4J_q}}. \tag{16}$$

The first quantity is the integrated squared error (ISE), or squared L^2 distance; its numerical value depends on coordinate units and scaling. The adopted Cauchy–Schwarz convention uses the squared normalized overlap and is therefore twice the alternative convention without the squares. Both Cauchy–Schwarz conventions are symmetric and nonnegative. JR_2 is the symmetric order-2 Jensen–Rényi entropy gap; because quadratic Rényi entropy is not generally concave, this functional need not be nonnegative. The total square loss $D_{\mathrm{tSL}}(p : q)$ is a nonnegative total-Bregman divergence oriented at q and is generally asymmetric. Equation (16) follows the Hilbert-space convention of [17]; variables with physical units therefore require nondimensionalization with respect to a chosen reference scale before evaluation. These distinctions prevent the four functionals from being conflated with metrics while making them available for assessing propagation and reduction results.

IV. APPROXIMATION AND REDUCTION

A. Nonlinear Propagation

For a nonlinear map $f : \mathbb{R}^{d_X} \to \mathbb{R}^{d_Y}$, the pushforward law of $Y = f(X)$ need not be a finite GMM and may lack a density with respect to output-space Lebesgue measure. GMTORCH constructs a fitted surrogate with density $\widehat{p}_Y$:

$$\begin{aligned} Y &= f(X), \qquad x_n \overset{\mathrm{iid}}{\sim} p_X, \\ y_n &= f(x_n), \quad n = 1, \ldots, N, \\ \widehat{p}_Y(y) &= \mathcal{G}_Y(y|\hat{g}_Y). \end{aligned} \tag{17}$$

Here, $\hat{g}_Y$ denotes the parameters obtained by fitting a K_Y-component mixture to $\{y_n\}_{n=1}^N$ using the regularized EM procedure described in Section II. The user selects N and K_Y; input and output dimensions may differ. When the pushforward has a density, $\widehat{p}_Y$ provides a density approximation. For singular outputs, such as (X, X^2) for nondegenerate scalar Gaussian X, the fitted positive-definite covariances spread probability off the exact support. Such singular laws cannot be approximated in total variation by these full-dimensional mixtures. The fit depends on sampling, mixture order, regularization, initialization, and local optimization; this path is neither exact nor differentiable through sampling and refitting.

Figure 2 illustrates this sample–transform–refit path: fitting supplies a reusable finite-mixture surrogate, without implying exact preservation of the transformed law or its support.

B. Mixture Reduction

After component-expanding operations, reduction replaces an input mixture by a chosen lower-order approximation. The currently provided strategy samples the input and fits a K_{target}-component mixture by EM, using the updates in Section II (Eqs. 3–4; [6]). It reuses the same approximation mechanism as nonlinear propagation and consequently has the same stochastic and optimization limitations.

The optional component limit (`max_components` in the listing below) on an exact propagation chain is a result-size

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

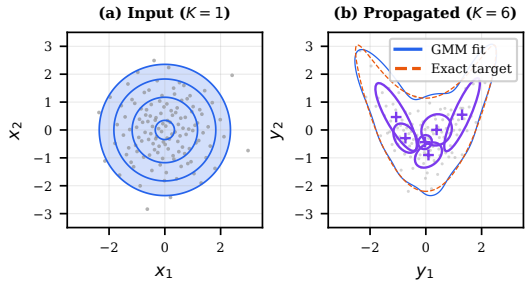


Fig. 2. Nonlinear propagation of $X \sim \mathcal{N}(0, I_2)$ under $f(x_1, x_2) = (x_1, [x_2 + 0.8(x_1^2 - 1)]/\sqrt{2.28})$. (a) Input density and Sobol display points; (b) the six-component GMM fitted by `propagate_gmm` to 20,000 i.i.d. transformed samples. Fitted and exact target contours are shown at the common density level 0.01; the analytical target is used only as a reference. Purple crosses mark component means; ellipses show one-standard-deviation covariance contours. Gray Sobol points (128, seed 202) are for visualization only. The fitted GMM is an approximate reusable representation of the nonlinear pushforward.

limit: it aborts after a step whose result exceeds the specified count, rather than serving as a pre-allocation memory guard. Reduction instead constructs an approximate lower-order mixture.

V. Implementation

The `GaussianMixture` object stores weights, means, and full covariance matrices as PyTorch tensors. Mixtures may be constructed directly or produced by `GaussianMixtureEstimator`; subsequent exact and approximate operations consume the same representation. Device and floating-point type therefore follow PyTorch tensor semantics without conversion to a second probabilistic data structure.

The component-stratified Sobol sampler allocates the requested sample count among mixture components according to their weights using largest-remainder rounding, and by default uses a separately scrambled Sobol sequence for each component; realized proportions can therefore differ slightly from weights. The nonlinear propagation procedure in (17) instead uses i.i.d. pseudorandom samples.

The implementation uses Cholesky factorizations and triangular solves rather than explicit covariance inverses in conditioning, density products, and Gaussian overlap calculations, with log-domain normalization of EM responsibilities and conditional and product weights, and log-domain accumulation of overlap sums. Pairwise product and overlap kernels are blocked to avoid materializing an additional full $K_X \times K_Y \times d \times d$ intermediate covariance tensor. Exact fusion still returns $K_X K_Y$ covariance matrices, so blocking does not remove the combinatorial output growth.

VI. Verification and Examples

A. Verification of Exact Operations

The validation suite contains reference checks that compare exact operations with independently assembled formulas. Affine maps are checked against per-component matrix operations; independent arithmetic is checked against all component pairs; and density-product fusion is checked both against direct precision formulas and against the normalized product evaluated at test points. Tests also verify that changing the computational block size leaves weights, means, and covariances unchanged within the configured numerical tolerances. Composed chains are compared with the same operations called individually, thereby checking orchestration separately from the underlying formulas.

B. Sequential Localization Update

Figure 3 illustrates one single-epoch robot-localization update. Panel (a) is the three-component robot self-perception prior; panels (c), (d), and (f) are single-Gaussian camera likelihood factors, and panel (e) is the resulting posterior. Reports follow $z_j = x + \epsilon_j$ with conditionally independent $\epsilon_j \sim \mathcal{N}(0, R_j)$. Each fixed R_j is a nominal-range local surrogate computed from the original prior mean $\bar{x}$ before sampling: for camera c_j, let $r_j = \|\bar{x} - c_j\| > 0$, $u_j = (\bar{x} - c_j)/r_j$, and v_j its perpendicular unit vector. With r_j in metres, the standard-deviation coefficients 0.08 and 0.04 are in metres, 0.025 is in inverse metres, and 0.05 is dimensionless; the resulting covariance is in square metres:

$$R_j = \left[0.08^2 + (0.025\, r_j^2)^2\right] u_j u_j^\top + \left[0.04^2 + (0.05\, r_j)^2\right] v_j v_j^\top. \tag{18}$$

The normalized spatial likelihood is $p(z_j \mid x) = \mathcal{N}(x \mid z_j, R_j) = \mathcal{N}(z_j \mid x, R_j)$. Because R_j is frozen independently of candidate x, this likelihood is Gaussian and permits analytic Gaussian fusion; recomputing a state-dependent covariance would generally lose that form. This illustrative additive error model is not a physically calibrated camera model. Conditional independence permits sequential application of the normalized density product (`BayesianFusion`) to the prior:

```
after_camera_1 = closed_form_propagation(
    predicted_prior,
    [BayesianFusion(camera_1)], max_components=3)
after_camera_2 = closed_form_propagation(
    after_camera_1,
    [BayesianFusion(camera_2)], max_components=3)
posterior = closed_form_propagation(
    after_camera_2,
    [BayesianFusion(camera_3)], max_components=3)
```

Each output feeds the next update, retaining three components at each step; no reduction is applied. This is an illustrative modeled update, not a validation of tracking performance. The map, field-of-view, ray, and occlusion overlays provide geometric context only and do not truncate the full-support GMM likelihood factors.

C. Nonlinear Propagation Example

Figure 2 illustrates an approximate finite-mixture fit to a nonlinear pushforward, with $N = 20{,}000$ and $K_Y = 6$. The fit uses the estimator defaults, with seed 123 and float64 explicitly set; sampling uses seed 42. Division by $\sqrt{2.28}$ gives the second output coordinate unit variance. Since f is invertible, its exact target density, with $s = \sqrt{2.28}$, is

$$p_Y(y) = \frac{s}{2\pi} \exp\left[-\frac{1}{2}\left(y_1^2 + [sy_2 - 0.8(y_1^2 - 1)]^2\right)\right]. \tag{19}$$

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

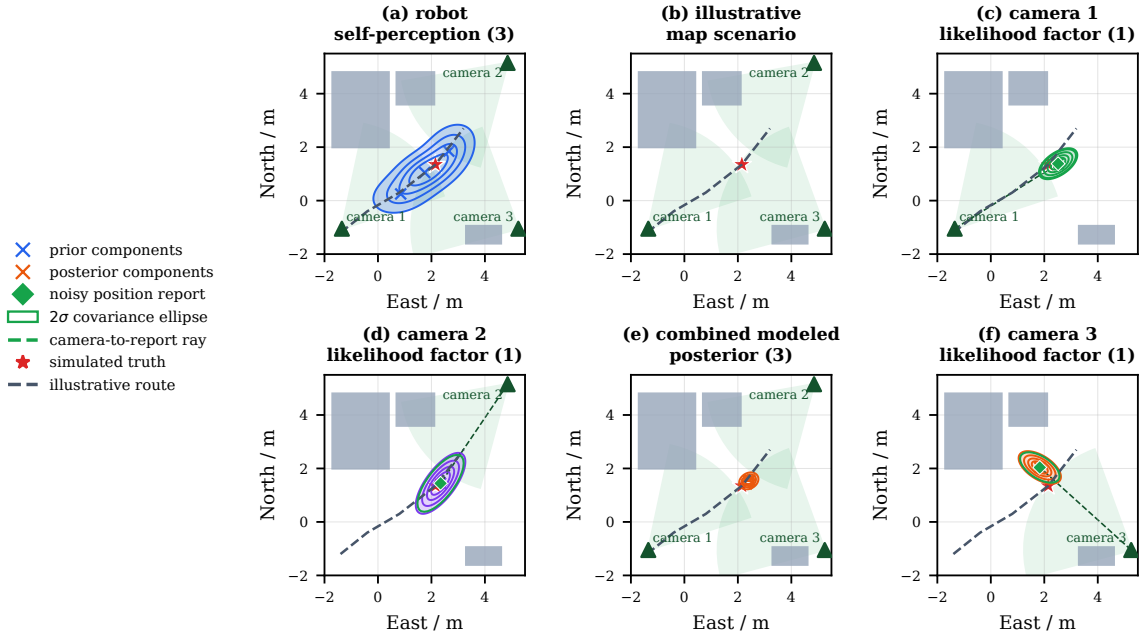


Fig. 3. Illustrative single-epoch urban robot-localization scenario and GMTORCH localization update. Each perception panel overlays the common map and building/occlusion context; camera panels additionally show the corresponding field of view, noisy report, two-standard-deviation covariance ellipse, and ray. Context does not affect or truncate the full-support likelihoods. The displayed GMMs use a three-component robot self-perception prior and three conditionally independent single-Gaussian likelihood factors. Sequential normalized products retain three components; contours are normalized relative to each mixture's maximum, crosses mark component means, diamonds mark noisy reports, and the red star marks the simulated true robot position.

For the regular input-coordinate domain $D = [-7, 7]^2$, let $\Omega = f(D)$; the finite-domain error is $\mathrm{ISE}_\Omega = \frac{1}{s}\int_D [p_Y(f(x)) - \widehat{p}_Y(f(x))]^2\, dx$, where $1/s$ is the transformation's Jacobian determinant. Using these settings, composite trapezoidal quadrature on an 801^2 grid gives $\mathrm{ISE}_\Omega = 9.3841 \times 10^{-4}$ and $J_\Omega = \frac{1}{s}\int_D p_Y(f(x))^2\, dx \approx 0.120159$; the corresponding relative L^2 error $\sqrt{\mathrm{ISE}_\Omega / J_\Omega}$ is 8.84%. The relative ISE change from the 401^2 grid is 3.04×10^{-10}. This refinement diagnostic is not an error bound, and the finite-domain calculation does not assess tails outside Ω.

VII. Discussion

The central benefit of the calculus is continuity of representation. A mixture-valued result remains a distribution rather than becoming an operation-specific parameter array: it can be evaluated, sampled, summarized, conditioned, or used as input to another calculation. This continuity is independent of whether the initial mixture originated from data or from a model, and it permits exact and approximate steps to coexist without changing the represented type of uncertain quantity.

In measurement-system modeling [4], paired observations of a measurand X and observable Y can be used to estimate a joint mixture. Once this model is represented in GMTORCH, an observation $Y = y^*$ produces the complete probabilistic measurement result $p(X \mid Y = y^*)$ by exact conditioning under the fitted joint-mixture model. The workflow illustrates the intended separation: data are used to construct an uncertain quantity once, after which algebraic operations continue the calculation. Likewise, application-specific constructions such as geometric mixture representations [20] can enter the same calculus without first being re-estimated from samples.

This benefit is accompanied by a trade-off between analytical closure and representation complexity. Marginalization, conditioning, and affine maps preserve both the finite GMM family and its component count. Independent arithmetic and normalized density products also remain exact, but generate pairwise component combinations. Exactness therefore does not imply tractability over long chains. Reduction can control this growth only by introducing an approximation. Device acceleration and blocked computation can change resource use, but not this combinatorial property.

The meaning of a calculation also depends on its probabilistic model. The sum and difference formulas require independence; correlated inputs cannot be combined from their marginal mixtures alone. A normalized density product is a Bayesian update only when one factor is proportional to the likelihood of the observation for the same latent variable. Making these assumptions explicit is necessary for treating GMMs as uncertain values rather than as interchangeable parameter collections.

Nonlinear propagation and the provided reduction strategy currently rely on sampling and EM refitting. Their error depends on finite sampling, the chosen output order, initialization, and local optimization. The current reduction is consequently general but not specialized to preserving a particular divergence or moment criterion. More direct parameter-based reduction and deterministic approximation schemes are natural extensions.

Further limitations follow from the use of full positive-

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

definite covariance matrices and explicit finite mixtures. Storage and factorization costs grow with dimension, and binary operations can become impractical as their component products accumulate. Future work should therefore emphasize adaptive complexity control, specialized reduction, additional covariance parameterizations, and alternative nonlinear propagation methods. The paper does not claim new formulas for the individual GMM operations; its scope is their consistent implementation as a calculus for continued uncertain computation.

VIII. Conclusion

We presented GMTORCH, a PyTorch framework for computing with uncertain quantities represented by finite GMMs. Construction and estimation provide entry points to the representation; mixture-valued results remain complete distributions that can participate in subsequent calculations.

The framework integrates established analytical operations—including marginalization, conditioning, affine maps, independent arithmetic, and normalized density products—with explicit nonlinear propagation and mixture reduction. The former retain the finite-mixture representation exactly under their stated assumptions, whereas the latter use sampling and refitting when closure or tractable component counts are lost. In this way, exactness, probabilistic assumptions, and representation growth remain visible across a calculation rather than being hidden by a change of data model.

GMTORCH thus contributes a consistent software realization of GMM uncertainty calculus rather than new formulas for its individual operations. Its common representation connects model-based construction, data-based estimation, algebraic computation, and controlled approximation using standard PyTorch tensor semantics. The documentation is publicly available at `https://gmtorch.github.io`, and the package is distributed through the `Python Package Index` (`PyPI`) at `https://pypi.org/project/gmtorch`.

IX. Declaration on the Use of Artificial Intelligence

GPT-6 Astra and GPT-6.1 Sol were used to assist in preparing the example code and visualizations and in the editorial revision of the manuscript. The authors retain responsibility for the final content.

Acknowledgment

This work was funded by Project SFB 1574 “A Circular Factory for the Perpetual Product”, Deutsche Forschungsgemeinschaft (DFG), Project ID 471687386.

References

[1] Y. Lu, A. Stuart, and H. Weber, “Gaussian approximations for probability measures on $\mathbb{R}^d$,” *SIAM/ASA Journal on Uncertainty Quantification*, vol. 5, no. 1, pp. 1136–1165, 2017.

[2] G. J. McLachlan and D. Peel, *Finite Mixture Models*, ser. Wiley Series in Probability and Statistics. New York, NY, USA: John Wiley & Sons, 2000.

[3] T. J. Sullivan, *Introduction to Uncertainty Quantification*, 1st ed., ser. Texts in Applied Mathematics. Cham, Switzerland: Springer International Publishing, Dec. 2015.

[4] A. Darijani, J. Beyerer, Z. S. H. Nasrollah, L. Hoffmann, and M. Heizmann, “Comprehensive description of uncertainty in measurement for representation and propagation with scalable precision,” 2026. [Online]. Available: https://arxiv.org/abs/2603.20365

[5] 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: Statistical Methodology*, vol. 39, no. 1, pp. 1–22, 1977.

[6] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information Science and Statistics. Springer-Verlag New York, 2006.

[7] I. Sobol’, “On the distribution of points in a cube and the approximate evaluation of integrals,” *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86–112, 1967. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0041555367901449

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html

[9] J. Schreiber, “Pomegranate: Fast and flexible probabilistic modeling in python,” *Journal of Machine Learning Research*, vol. 18, no. 164, pp. 1–6, 2018. [Online]. Available: https://jmlr.org/papers/v18/17-636.html

[10] J. Schreiber and contributors, “pomegranate general mixture model documentation,” 2026, accessed: 2026-09-09. [Online]. Available: https://pomegranate.readthedocs.io/en/latest/tutorials/B_Model_Tutorial_2_General_Mixture_Models.html

[11] M. Varrone and contributors, “TorchGMM: Gaussian mixture models in PyTorch,” 2026, version 0.1.4. Accessed: 2026-09-09. [Online]. Available: https://github.com/CSOgroup/torchgmm

[12] O. Borchert, “PyCave: Machine learning on large data sets,” 2022, accessed: 2026-09-09. [Online]. Available: https://github.com/borchero/pycave

[13] A. Fabisch, “gmr: Gaussian mixture regression,” *Journal of Open Source Software*, vol. 6, no. 62, p. 3054, 2021.

[14] T. van den Berg, “cgmm: Conditional Gaussian Mixture Models for Python,” https://github.com/sitmo/cgmm, 2024, version 0.5.0.

[15] B. Delhaisse, L. Rozo, and D. G. Caldwell, “PyRoboLearn: A Python framework for robot learning practitioners,” in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 2020, pp. 1348–1358. [Online]. Available: https://proceedings.mlr.press/v100/delhaisse20a.html

[16] B. Delhaisse and contributors, “PyRoboLearn gaussian-mixture implementation,” 2020, source file, accessed 2026-09-10. [Online]. Available: https://github.com/robotlearn/pyrobolearn/blob/master/pyrobolearn/models/gmm/gmm.py

[17] F. Nielsen, “Closed-form information-theoretic divergences for statistical mixtures,” in *Proceedings of the 21st International Conference on Pattern Recognition*. IEEE, 2012, pp. 1723–1726. [Online]. Available: https://www2.sonycsl.co.jp/person/nielsen/CauchySchwarzDivergence-ICPR2012_accepted.pdf

[18] F. Wang, T. Syeda-Mahmood, B. C. Vemuri, D. Beymer, and A. Rangarajan, “Closed-form jensen–rényi divergence for mixture of gaussians and applications to group-wise shape registration,” in *Medical Image Computing and Computer-Assisted Intervention*. Springer, 2009, pp. 648–655.

[19] M. Liu, B. C. Vemuri, S.-i. Amari, and F. Nielsen, “Shape retrieval using hierarchical total bregman soft clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2407–2419, 2012.

[20] B. Stratmann, E. Soufian, J. Beyerer, and A. Darijani, “A geometric gaussian mixture representation of curves and trajectories in euclidean space,” in *2026 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, Pilsen, Czechia, Sep. 2026, accepted for publication.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.