Ultralow-Power Localization of Insect-Scale Drones: Interplay of Probabilistic Filtering and Compute-in-Memory

Priyesh Shukla, *Graduate Student Member, IEEE*, Ankith Muralidhar, Nick Iliev, Theja Tulabandhula, Sawyer B. Fuller, and Amit Ranjan Trivedi, *Senior Member, IEEE*

***Abstract*— We propose a novel compute-in-memory (CIM)-based ultralow-power framework for probabilistic localization of insect-scale drones. Localization is a critical subroutine for path planning and rotor control in drones, where a drone is required to continuously estimate its pose (position and orientation) in flying space. The conventional probabilistic localization approaches rely on the 3-D Gaussian mixture model (GMM)-based representation of a 3-D map. A GMM model with hundreds of mixture functions is typically needed to adequately learn and represent the intricacies of the map. Meanwhile, localization using complex GMM map models is computationally intensive. Since insect-scale drones operate under extremely limited area/power budget, continuous localization using GMM models entails much higher operating energy, thereby limiting flying duration and/or size of the drone due to a larger battery. Addressing the computational challenges of localization in an insect-scale drone using a CIM approach, we propose a novel framework of 3-D map representation using a harmonic mean of the "Gaussian-like" mixture (HMGM) model. We show that *short-circuit current* of a multiinput floating-gate CMOS-based inverter follows the harmonic mean of a Gaussian-like function. Therefore, the likelihood function useful for drone localization can be efficiently implemented by connecting many multiinput inverters in parallel, each programmed with the parameters of the 3-D map model represented as HMGM. When the depth measurements are projected to the input of the implementation, the summed current of the inverters emulates the likelihood of the measurement. We have characterized our approach on an RGB-D scenes dataset. The proposed localization framework is ∼25× energy-efficient than the traditional, 8-bit digital GMM-based processor paving the way for tiny autonomous drones.**

***Index Terms*— Compute-in-memory (CIM), insect-scale drones, probabilistic localization.**

Manuscript received February 14, 2021; revised May 24, 2021 and June 24, 2021; accepted July 10, 2021. Date of publication August 5, 2021; date of current version January 17, 2022. This work was supported in part by Intel and CAREER Award #2046435 by the National Science Foundations (NSF). *(Corresponding author: Priyesh Shukla.)*

Priyesh Shukla, Ankith Muralidhar, Nick Iliev, and Amit Ranjan Trivedi are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: pshukl23@uic.edu).

Theja Tulabandhula is with the Department of Information and Decision Sciences, University of Illinois at Chicago, Chicago, IL 60607 USA.

Sawyer B. Fuller is with the Department of Mechanical Engineering, University of Washington, Seattle, WA 98115 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2021.3100252.

Digital Object Identifier 10.1109/TVLSI.2021.3100252

I. Introduction

SPATIALLY intelligent devices can autonomously traverse and explore their application domain, thereby opening intriguing prospects for sensors and embedded systems. For example, in an agriculture field, flying cameras can locate infected plants to prevent disease spread [3], [4]; flying gas sensors can identify gas leaks in an industrial plant [5], [6]; during earthquakes, autonomous cameras can assist in search and rescue missions by navigating through building debris [7], [8]; and inside a home, a flying camera can actively monitor fire breakouts and intruders [9], [10]. Thus, most embedded systems, enlivened by spatial intelligence, can have dramatically elevated use cases. However, the flying devices must also be small enough to be nonintrusive to people in the flying spaces and be able to navigate through constricted spaces. Recently, impressive progress has been made in miniaturizing drones. The so-called *insect-scale drones* studied in [1] and [11]–[13] weigh less than a gram and are smaller than a penny as shown in Fig. 1. Meanwhile, since an insect-scale drone can only carry the payload of a tiny battery, minimizing power dissipation of on-board processing for spatial intelligence becomes extremely critical [14]. Our motivation for exploring ultralow-power on-board processing of robotic tasks such as localization comes from recent insect-scale drone designs [1], [2], [15], [16] where power requirements for flights have already scaled down to levels comparable to necessary for drone autonomy. For example, RoboFly [1] weighs only 100 mg with a wingspan of 30 mm and requires only ∼50 mW power for hovering. Meanwhile, current custom-designed state-of-the-art chips require tens of milliwatt power for drone localization and odometry [14], [17]. In Fig. 1, we consider further scaling trends for insect-scale drones considering two approaches. In the first mass scaling approach, only drone weight is scaled down. For example, lighter drone materials such as graphene can be used [2]. In the second mass and wing-span scaling approach, both weight and wing-span are proportionally scaled down. Considering simple physics laws [18], the necessary power for drone hovering scales down as $\sim m^{3/2}/r$ where m is the drone mass and r is its wingspan radius. In the figure, with 50% reduction in RoboFly's weight, the necessary hovering power reduces to ∼20 mW. Therefore, to support the continued scaling of insect-scale drones, disruptive approaches

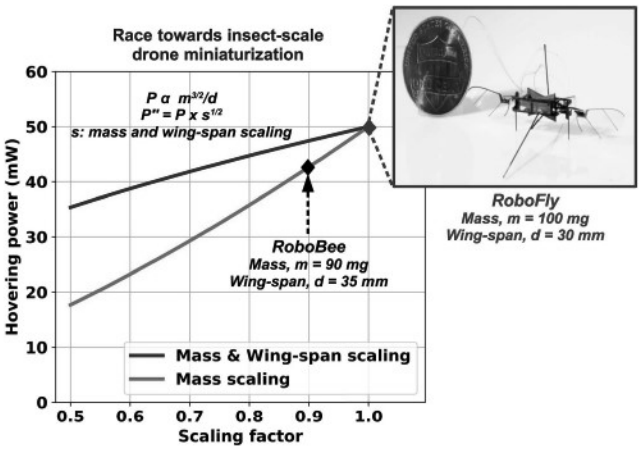

Fig. 1. Power with scaling down of drone's mass and dimensions. The insect-scale drones, RoboFly [1] (inset figure), and RoboBee [2] weigh less than a gram and are smaller than a penny.

to support ultralow-power localization will be necessary. While the cloud can be leveraged to circumvent the above on-board processing challenges, in many applications, reliable continuous connectivity to the cloud cannot be assumed. Cloud-based control may also suffer from unpredictable latency. Moreover, continuous transmission of drone's video stream to cloud itself is energy hungry [19]. Hence, on-board ultralow-power processing capacity for self-navigation is indispensable.

The *most basic operation* for self-navigation is to determine the position and orientation (i.e., pose) of a drone during its flight. Path planning objectives such as motion tracking and obstacle avoidance require one to continuously assess the drone's pose. Therefore, drone localization must also be evaluated in real-time. Additionally, since the flying space of a drone is dynamic, drone localization must be robust against dynamic variations in the flying space. For example, in an indoor application, there will be movement of people and changes in lighting conditions. Localization of a drone must be robust against these variations. Bayesian probabilistic inference is a promising approach for enhancing the robustness of inference by extracting both the prediction and the *confidence* of prediction [20]–[22]. A probabilistic framework for drone localization models the likelihood of measurement against a hypothesized drone's pose model [23]. The posterior probability of drone's pose can be compared against competing hypotheses to identify the most likely one. The likelihood models can also estimate the prediction confidence based on the variance of the prediction. A high confidence prediction will have low variance in estimates and *vice versa*. Nonetheless, current state-of-the-art probabilistic localization models are quite computationally intensive. While predictive robustness of a probabilistic localization is highly desirable in an insect-scale drone, incorporating the same is quite challenging due to limited computational resources as well as the need to predict in the real-time. Overcoming the above challenges, this work makes the following key contributions:

1) We propose a novel representation of 3-D maps using a harmonic mean of a "Gaussian-like" mixture (HMGM) model. We show that, unlike a Gaussian mixture model (GMM), HMGMs can be computed with much higher efficiency and enable a similar localization accuracy as GMM. We also discuss an expectation maximization (EM)-based learning procedure for HMGM's extraction.
2) We use the key property that the short-circuit current of a multi-input floating gate (FG) CMOS inverter circuit follows the harmonic mean of Gaussian-like (HMG) functions. Therefore, by emulating the key computing kernels in our localization approach directly at the transistor level, we are able to minimize computing workload and data movements in the processing, and thereby energy demand for the computing. Using the proposed implementation illustrated in Fig. 2, the localization model is evaluated with high parallelism, and therefore, timing and energy constraints for real-time predictions that were hard to achieve are now feasible.
3) We have characterized our localization approach using an RGB-D dataset for 12 scenes. The maximum localization error in our hardware-based approach is 1.56 m from the ground truth for Scene-08 which is only slightly worse than software-based error, viz. 1.39 m. Our localization framework, meanwhile, is ultralow power. Our approach consumes ∼2.1 pJ when processing a depth pixel in 20 ns to compute log-likelihood of a hypothesized pose in particle-filtering (PF). Comparatively, a customized 8-bit digital processor consumes ∼358 pJ energy which is 170× higher than our approach. We have also rigorously analyzed the impact of process variability and limited precision in our design and document methodologies to mitigate their impact.

II. Overview of Probabilistic Localization

Localization of an autonomous mobile agent invites uncertainties in its pose estimates. A probabilistic localization framework can rigorously account for such modeling and measurement uncertainties to not only predict the estimated pose but also the *confidence* of prediction, which is expressed by the variance of the predicted estimate. Extracting prediction confidence along with the prediction itself allows for a *risk-aware* navigation, wherein the drone can seek additional measurements if the prediction confidence is low.

Using Bayes' rule, the posterior probability of pose ($\mathbf{x}$) of a drone is given by

$$P(\mathbf{x}|\mathbf{z}) = \frac{P(\mathbf{z}|\mathbf{x})P(\mathbf{x})}{P(\mathbf{z})}. \tag{1}$$

Here, $P(\mathbf{z}|\mathbf{x})$ is the likelihood of measurement $\mathbf{z}$ and $P(\mathbf{x})$ is the prior probability of pose belief. A maximum likelihood estimate (MLE) of drone's pose maximizes the likelihood of the current measurement, that is,

$$\mathbf{x}_{\text{MLE}} = \text{argmax } P(\mathbf{z}|\mathbf{x}). \tag{2}$$

Meanwhile, the maximum *a posteriori* estimate (MAP) of pose searches for the estimate that maximizes posterior probability of the pose, that is,

$$\mathbf{x}_{\text{MAP}} = \text{argmax } P(\mathbf{z}|\mathbf{x})P(\mathbf{x}). \tag{3}$$

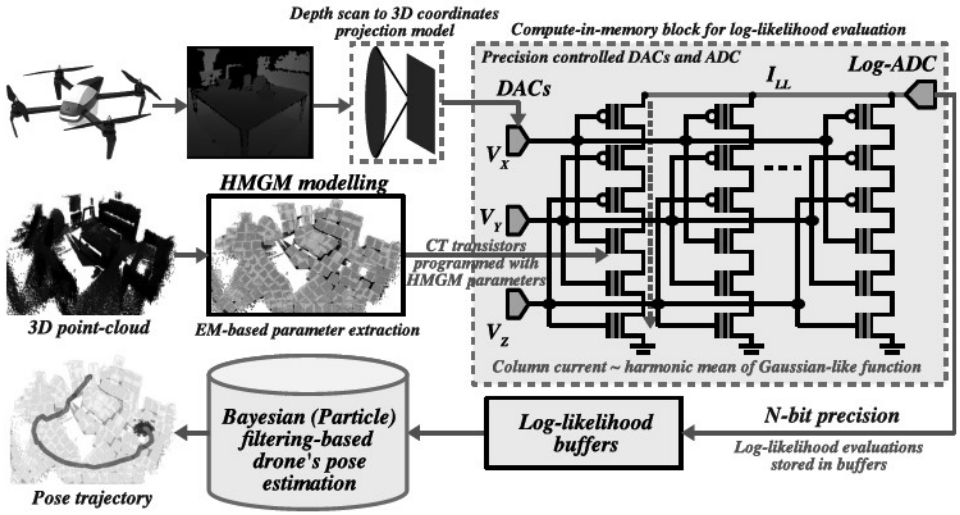

Fig. 2. CIM-based ultralow-power Monte Carlo localization framework for insect-scale drones. The framework inputs sequential depth projections. CIM is programmed with HMGM model parameters. CIM computes likelihood of depth measurements corresponding to hypothesized pose which is further used in particle filtering (PF) to predict the pose of drone.

Under global localization, when the drone is completely uncertain about its pose, MAP and MLE are equivalent. Otherwise, MAP estimate allows for a systematic way to account for prior belief. Using probabilistic localization, a drone can also recursively integrate sequential measurements in a systematic framework to reduce its estimation uncertainties. *Bayesian filtering* [24] is a well-established localization method for this purpose where the belief of drone's pose can be estimated using a recursive flow of Bayes' rule as

$$\overline{\text{bel}}(\mathbf{x_t}) = \sum_{x_{t-1}} P(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}})\text{bel}(\mathbf{x_{t-1}}) \tag{4a}$$

$$\text{bel}(\mathbf{x_t}) = \eta P(\mathbf{z_t}|\mathbf{x_t})\overline{\text{bel}}(\mathbf{x_t}). \tag{4b}$$

Here, $\overline{\text{bel}}(\mathbf{x_t})$ in (4a) is the drone's new pose belief at iteration "t." $\text{bel}(\mathbf{x_{t-1}})$ is the pose belief at previous iteration "$t-1$." $\mathbf{u_t}$ is the control input to the drone. $P(\mathbf{x_t}|\mathbf{u_t}, \mathbf{x_{t-1}})$ is the probabilistic motion model exploiting Markov assumption where belief of $\mathbf{x_t}$ depends only on the previous state's belief, and the most recent control signal $\mathbf{u_t}$. Equation (4b) is the correction on the new pose belief based on the sensor measurements.

Particle filtering [25], [26] allows a practical implementation of Bayesian filtering under arbitrary likelihood and belief function profiles by using a Monte Carlo method. Instead of relying on analytically defined likelihood and belief functions, the method considers a set of hypotheses (aka particles) sampled based on the underlying density functions. Each particle has an associated weight index which is updated based on the measurements following the above equations. Unlikely particles with low likelihood are sequentially filtered out with each measurement. Various uncertainties, such as in drone's motion control, are accounted for by regenerating a new particle set from the current, where children particles represent stochastic deviations corresponding to motion control uncertainties.

Although Bayes' rule-based probabilistic localization allows a systematic framework to seamlessly integrate single-shot MLE/MAP estimates of drone poses as well as sequential localization steps (Bayesian filtering) as discussed above, the procedure can be quite computationally intensive. The most dominant complexity is the estimation of likelihood term $P(\mathbf{z}|\mathbf{x})$ itself since the measurements (images/depth maps) are high dimensional and models correlating drone's pose with the measurements can be quite complex depending on the size and intricacies of the 3-D map.

A popular approach to extract $P(\mathbf{z}|\mathbf{x})$ is by modeling the 3-D map using a GMM [27]–[29]. Using 3-D scanning devices, such as Microsoft Kinect [30], [31], a point-cloud data-based 3-D map of the domain can be captured. A GMM is synthesized to model the point-cloud data which essentially represents the density of "matter" in the 3-D map. To adequately model intricacies of the 3-D map, a sufficient number of mixture functions in the GMM are necessary. During flight, depth sensors in a drone capture a depth map of its current observation. Although depth sensing is energy expensive, recent works such as [32] have shown energy-efficient, small form-factor depth sensors, suited for insect-scale drones. Based on the current pose belief, the scan $\mathbf{z}$ of N nonzero depth map pixels $\{z_1, z_2, \ldots, z_N\}$ is projected to 3-D via the camera's projection model

$$\mathbf{C_c} = \begin{bmatrix} o_x + f\dfrac{\mathbf{C_x}}{\mathbf{C_z}} \\ o_y + f\dfrac{\mathbf{C_y}}{\mathbf{C_z}} \end{bmatrix} \tag{5a}$$

$$\mathbf{C_w} = [\mathbf{C_c} - \mathbf{T_{cw}}]\mathbf{R_{cw}^{-1}}. \tag{5b}$$

Here, $\mathbf{C_x}$, $\mathbf{C_y}$, and $\mathbf{C_z}$ are the point coordinates in 3-D space. Their 2-D perspective projection in the camera frame is $(f(\mathbf{C_x}/\mathbf{C_z}), f(\mathbf{C_y}/\mathbf{C_z}))$. $\mathbf{C_c}$ is the depth pixel coordinate matrix in camera frame mounted on the insect-scale drone. f is the focal length, and o_x and o_y are optical offsets. $\mathbf{T_{cw}}$ and $\mathbf{R_{cw}}$ are translation and rotation matrices, respectively, transforming depth scans in camera frame to the world frame based on

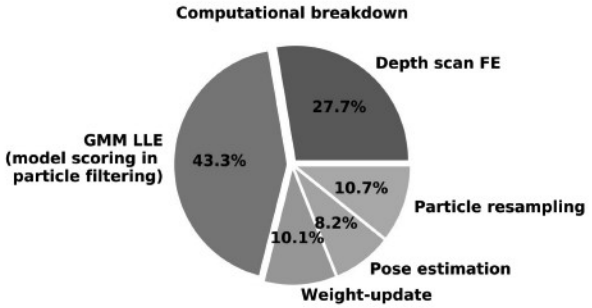

Fig. 3. Pie chart of the breakdown of number of computations (in log-scale) clearly illustrating the dominance of log-likelihood evaluations in the entire particle filtering flow.

the pose belief. Thereby, $P(\mathbf{z}|\mathbf{x})$ is estimated by evaluating $p_{\text{GMM}}(\mathbf{z}_{\text{proj}}(\mathbf{x}, \mathbf{z}); \boldsymbol{\Theta})$, where $\mathbf{z}_{\text{proj}}(\cdot)$ is the projection function of measurements $\mathbf{z}$ to 3-D space based on pose belief $\mathbf{x}$. $\boldsymbol{\Theta}$ are the GMM parameters.

In a typical evaluation of pose localization, the likelihood $p_{\text{GMM}}(\mathbf{z}_{\text{proj}}(\mathbf{x}, \mathbf{z}); \boldsymbol{\Theta})$ is computed for hundreds of nonzero depth pixels and over hundreds of mixture functions in the GMM model. Therefore, the computational workload of $P(\mathbf{z}|\mathbf{x})$ is excessive. Moreover, during a flight, the drone needs to continuously localize itself; hence, the timing constraints on $P(\mathbf{z}|\mathbf{x})$ are also stringent. In Section III, we discuss a novel approach to considerably minimize the energy demand for measurement likelihood evaluation which eventually leads to a viable low power probabilistic localization approach for insect-scale drones.

In Fig. 3, we show the computational breakdown of various steps in a Monte Carlo localization algorithm which are discussed elaborately in Section V. In the figure, we consider hundred pose-hypotheses (particles), hundred mixture components, and depth image of 640 × 480 pixel resolution for a 3-D localization test case. The computational breakdown is extracted by considering the total number of operations in each step, whereas each operation consists of an addition, subtraction, multiplication, division, or memory access. As evident in the figure, likelihood evaluations are the most computationally expensive step in a Monte Carlo localization algorithm. The workload of likelihood evaluation increases further with more mixture components (necessary for a more extensive flying space) and more particles (necessary for higher robustness). Therefore, in this work, we have mainly focused upon exploring alternative processing procedures to minimize the workload of likelihood evaluations.

III. Ultralow-Power Probabilistic Localization

Prior works [27]–[29] use a GMM to probabilistically represent a 3-D space, that is, model the density of matter in the 3-D map. However, evaluation of likelihood models based on GMM is computationally intensive. A dimension of each mixture function in GMM requires subtractions, multiplications, additions (to compute the exponent of a Gaussian function) as well as look-ups (to add exponents using a log-ADD table [33]–[35]). Since these operations are repeated on all dimensions, mixture functions as well as the hypotheses,

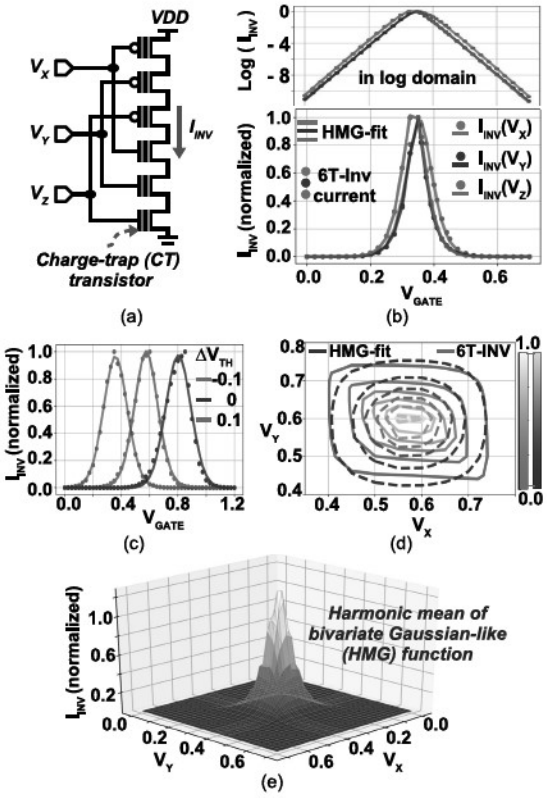

Fig. 4. (a) 6T-inverter circuit to emulate HMG function. (b) I_{INV} behaves Gaussian-like upon varying one of the input voltages—V_X, V_Y, or V_Z. (c) Programming I_{INV} peak by controlling VTH (ΔV_{TH} in volts). (d) and (e) Contour and surf plots with bivariate input (V_X, V_Y) control.

likelihood evaluation of a typical depth map may require tens of thousands of arithmetic operations and hundreds of memory look ups to evaluate the likelihood of a single measurement.

A. Mixture of HMG Functions

We discuss a novel representation of 3-D maps which can considerably simplify the computation of measurement likelihoods. Consider the six transistor inverter design controlled by three input voltages, V_X, V_Y, and V_Z, in Fig. 4(a). The transistors in the inverter comprise an FG to program threshold voltage (VTH) [36]–[38]. Therefore, by programming the charge density in the FG, the VTH of transistors can be programmed in a nonvolatile manner. In Fig. 4(b), the inverter current, I_{INV}, through series-connected transistors emulates a Gaussian-like function. The figure shows I_{INV} when varying one of the input voltage—V_X, V_Y, or V_Z—while keeping the others fixed.

Notably, the Gaussian-like nature of inverter's short-circuit current can be understood by representative transistor's current equations. In Fig. 4(b), close to the peak, inverter's short-circuit current is governed by equating nMOS and pMOS current in the saturation regime, that is,

$$I_{N,\text{SAT}} \approx \frac{\beta_n}{2} V_{dr,n}^2 (1 + \lambda_n V_{\text{OUT}}) \tag{6a}$$

$$I_{P,\text{SAT}} \approx \frac{\beta_p}{2} V_{dr,p}^2 (1 + \lambda_p (V_{\text{DD}} - V_{\text{OUT}})). \tag{6b}$$

Here, $V_{dr,n} = V_{\text{IN}} - V_{\text{TH},n}$ is nMOS's drive voltage and $V_{dr,p} = V_{\text{DD}} - V_{\text{IN}} - V_{\text{TH},p}$ is pMOS's drive voltage. $V_{\text{TH},n/p}$ is the VTH of nMOS/pMOS, V_{IN} is the input voltage to the inverter, λ is the channel length modulation (CLM) parameter, and β_n and β_p are the transconductance coefficients of nMOS and pMOS, respectively. Since $I_{N,\text{SAT}} = I_{P,\text{SAT}}$, V_{OUT}, and V_{IN} are related as

$$V_{\text{OUT}} \approx \frac{\beta_p V_{dr,p}^2(1+\lambda_p V_{\text{DD}}) - \beta_n V_{dr,n}^2}{\lambda_n \beta_n V_{dr,n}^2 + \lambda_p \beta_p V_{dr,p}^2}. \tag{7}$$

Therefore, inverter's short-circuit current near the peak is determined by applying V_{OUT} expression from (7) to (6a). Under the setting, $\beta_p = \beta_n$ and negligible CLM, inverter's current near the peak is given by

$$I_{\text{INV,near\_peak}} \approx \beta_n \frac{V_{dr,n}^2 V_{dr,p}^2}{V_{dr,n}^2 + V_{dr,p}^2}. \tag{8}$$

Therefore, the peak current voltage, V_{peak}, is centered at $V_{dr,n} = V_{dr,p}$, that is, $(V_{\text{DD}} + V_{\text{TH},n} - V_{\text{TH},p})/2$. At $V_{\text{TH},n} = V_{\text{TH},p}$, V_{peak} is at $V_{\text{DD}}/2$. V_{peak} can be modulated by Δ amount by programming inverter's VTHs as $V_{\text{TH},n} \rightarrow V_{\text{TH},n} + \Delta$ and $V_{\text{TH},p} \rightarrow V_{\text{TH},p} - \Delta$. Under these settings, the peak current magnitude is the same, and only V_{peak} modulates. Also, note that close to the peak, $I_{\text{INV,near\_peak}}$ has a quadratic dependence to V_{IN}, just like a Gaussian function has to the input variable near its peak.

Away from the peak, the inverter's current is governed by nMOS's (pMOS's) subthreshold current, that is,

$$I_{\text{INV,tail}} \approx I_{0,n} \exp\left(\frac{V_{dr,n}}{nV_T}\right). \tag{9}$$

Here, V_T is the thermal voltage and $I_{0,n}$ is the leakage current at $V_{dr,n} = 0$. Therefore, at the tail, the inverter's current decays exponentially. However, unlike in a Gaussian function, the exponent is linearly dependent on the argument variable (V_{IN}). Therefore, we can model the inverter's short-circuit current using a "Gaussian-like" function defined as

$$I_{\text{INV}} \approx I_{0,\text{INV}} \exp\left(-\frac{(V_{\text{IN}} - \mu)^2}{\sigma^2(\alpha + |V_{\text{IN}} - \mu|)}\right). \tag{10}$$

Here, μ is the input voltage where the inverter's short-circuit current peaks and depends on $V_{\text{TH},n}$ and $V_{\text{TH},p}$. α is a fitting parameter that models the transition from quadratic to exponential dependence of I_{INV} at increasing/decreasing V_{IN}. σ models the variance of I_{INV} and it depends on the thermal voltage and transistor's ideality factor n. Fig. 4(b) shows the correlation between the inverter current model in (10) and HSPICE-simulated inverter's short-circuit current in linear and log domains. Fig. 4(c) shows the modulation of μ by programming $V_{\text{TH},n}$ and $V_{\text{TH},p}$ in the inverter design.

Fig. 4(d) and (e) shows the contour and surface plots, respectively, of I_{INV} when varying two gate voltages at a time. At sufficiently higher supply voltage (VDD), under CLM, the peak current of an inverter linearly follows the voltage drop ΔV across the inverter, that is, $I_{\text{INV}} \propto \Delta$V. Therefore, when multiple inverters are connected in series, as in the 6-T inverter design in Fig. 4(a), the overall current of multiinput inverter follows $1/(1/I_{\text{INV},1} + 1/I_{\text{INV},2} + 1/I_{\text{INV},3})$ which is equivalent to the harmonic mean of currents from the constituent inverters. Under a multivariate control of inverter's current, that is, when more than one input voltages vary simultaneously, the characteristics of I_{INV} are, therefore, more closely represented by an HMG function in (10) where each constituent 1-D function is controlled by input V_X, V_Y, and V_Z. The column current in the design follows a harmonic mean of Gaussians since current through each stacked inverter is proportional to a Gaussian-like function. Therefore, vertically stacking them accumulates their currents through Kirchhoff's law, leading to harmonic mean of Gaussian dependence to the overall column current. The column current can be expressed as

$$I_{\text{INV}} \approx \frac{I_{0,\text{INV3D}}}{\sum_{i=X,Y,Z} \exp\left(\frac{(V_i - \mu_i)^2}{\sigma^2(\alpha + |V_i - \mu_i|)}\right)}. \tag{11}$$

Here, μ_X, μ_Y, and μ_Z are the respective mean of Gaussian-like functions which are controlled by the VTH of transistors connected with the input voltage V_X, V_Y, and V_Z, respectively.

Notably, compared to a multivariate Gaussian function, the implementation and evaluation of HMG in Fig. 4 is much simplified. While a digital implementation to evaluate multivariate Gaussian will require multiplier, adder, and look-up tables for exponential or log-ADD function [39]–[41], HMG in Fig. 4 is implemented using only six transistors and can be evaluated by applying analog voltages V_X, V_Y, and V_Z. Also, compared to [42], the model can be implemented using FG CMOS transistors and does not require modifications in transistor design and processes. As we will discuss later, for drone localization, V_X, V_Y, and V_Z will correspond to depth pixel's projection along 3-D spatial dimensions. A model with mixtures of HMG functions can also be simply implemented by connecting many multiinput inverters in Fig. 4(a) in parallel, each corresponding to a mixture component, and sharing their gate-input terminals. The total current from the parallel inverters will thus emulate the likelihood of measurements applied at the gates.

The likelihood of 3-D projection of depth scan, d_{meas}, for localization can, therefore, be computed using HMGM as

$$\ell(d_{\text{meas}}|\boldsymbol{\Theta}) = \sum_{i=1}^{N} \ln \sum_{j=1}^{M} \lambda_j I_{\text{INV}}\left(V_X^i, V_Y^i, V_Z^i; \mu_{\mathbf{j}}, \sigma_{\mathbf{j}}, \alpha_{\mathbf{j}}\right). \tag{12}$$

Here, $[V_X^i \; V_Y^i \; V_Z^i]$ is the 3-D projection of ith depth pixel to the world frame. Projection of depth scan to world frame was discussed in (5). $\boldsymbol{\Theta}$ represents the parameter set for HMGM, comprising of λ_i (mixing proportion), μ_i (mean), σ_i (standard deviation), and α_i (fitting parameter) for each respective mixture component i.

B. EM for HMGM

Parameters of HMGM ($\boldsymbol{\Theta}$) can be learned by adapting EM [43] procedure as shown in Algorithm 1. We begin with randomly initializing $\boldsymbol{\Theta}$. In the E-step, we compute the expected value of the log likelihood of 3-D map's point cloud dataset d_{pc}, that is, $\ell(d_{\text{pc}}|\boldsymbol{\Theta}_t)$. In the M-step,

Algorithm 1 EM for HMGM Representation

Input: $\mathbf{d_{pc}}$ (Point-cloud co-ordinates)
$\mathbf{\Theta^{init}}$ (Initial HMGM parameters)
(σ and α are tied to be same for all mixture components simplifying hardware complexity)
σ and α (predetermined from inverter characteristics)
τ (EM tolerance)
lr (Learning rate)
N (Total point-cloud data points for EM)
M (Number of HMGM components to learn)

while $\Delta\mathcal{E} > \tau$ **do**
  E-step:
  for i = *1 to N* **do**
    Compute:
    Membership probability, $\mathcal{M}(d^i_{pc}|\mathbf{\Theta^{(t)}})$:
    $\mathcal{M}_0 = 0$
    for j = *1 to M* **do**
      $\mathcal{M}_j = \mathcal{M}_{j-1} + \lambda_j I_{\mathrm{INV}}(d^i_{pc}; \mu_\mathbf{j}, \sigma, \alpha)$
      $\mathrm{I_{INV}}(\cdot) \rightarrow$ expressed in (11)
    end
    Expected log likelihood (proportional to I_{INV}**)**:
    $\ell(d^i_{pc}|\mathbf{\Theta^{(t)}}) = \ell(d^{i-1}_{pc}|\mathbf{\Theta^{(t)}}) + \ln \mathcal{M}(d^i_{pc}|\mathbf{\Theta^{(t)}})$
  end
  Expectation, $\mathcal{E}^{(\mathbf{t})} \equiv \ell(\mathbf{d_{pc}}|\mathbf{\Theta^{(t)}})$
  M-step:
  $\mathbf{\Theta^{(t+1)}} \equiv \arg\max_{\mathbf{\Theta}} \ell(\mathbf{d_{pc}}|\mathbf{\Theta^{(t)}})$
end
Output: $\mathbf{\Theta^{optimal}}$ (extracted HMGM parameters)

we compute optimal parameters $\Theta^{(t+1)}$ that maximizes the expected log likelihood formulated in the E-step, that is, $\mathbf{\Theta^{(t+1)}} \equiv \arg\max_{\Theta} \ell(d_{\mathrm{pc}}|\Theta_t)$. E-M steps are repeated until the log-likelihood of HMGM converges.

Under full programmability, variance $\sigma_\mathbf{j}$ of each mixture function can be learned independently. However, in our simplified implementation, the variance of each mixture function is dictated by the technology and transistor parameters (width, length, etc.). Conversely, the proposed scheme can easily implement a larger number of mixture functions with limited resource/power overhead. Therefore, we exploit a larger number of mixture components in our implementation to compensate for the lack of flexibility due to a tied variance. In Algorithm 1, only the mean and weights of mixture components are learned, therefore, and a predetermined σ based on 6T-inverter characteristics is used.

Fig. 5(a) illustrates the fitting of Scene 1 in RGB-D dataset [44] using our HMGM-based model implemented through inverter array. We use "model score" to quantify the goodness of fit. The model score ($\mathcal{MS}$) is evaluated as the average of HMGM function's value on point cloud data, $\mathbf{d_{pc}}$, in the negative log domain, that is,

$$\mathcal{MS} = -\frac{1}{N}\sum_{i=1}^{N} \ln \sum_{j=1}^{M} \lambda_j I_{\mathrm{INV}}(\mathbf{d_{pc}}; \mathbf{\Theta^{(t)}}). \tag{13}$$

A higher model score indicates a better representation of point cloud data. Fig. 5(b) shows that the model score for HMGM increases with EM iterations and then plateaus. Fig. 5(b) also shows the model score for GMM which shows similar fitting profile. In Fig. 5(c), the model score is plotted at increasing number of mixture components for both HMGM and GMM models. The model score for both models improves with more mixture components and then plateaus. Although

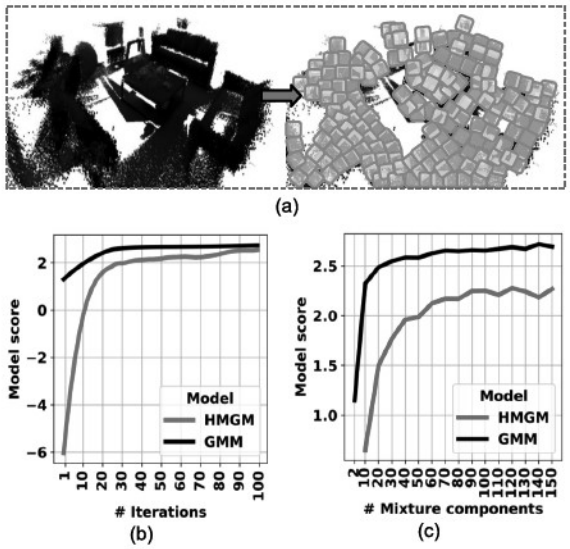

Fig. 5. (a) HMG mixture (HMGM) function representation of an indoor scene point cloud. (b) Model scores of HMGM and GMM during EM iterations to optimize model parameters over the point cloud. (c) Model scores improve with increase in the number of mixture components.

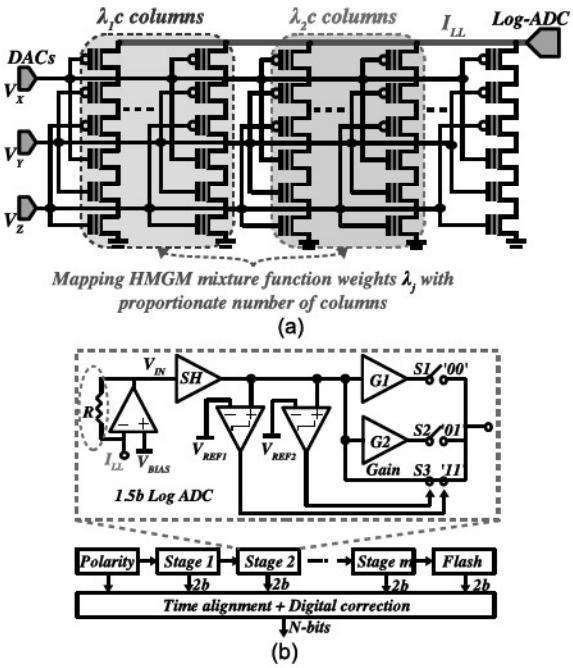

Fig. 6. (a) Architecture of the FG transistor-based CIM based on HMGM to evaluate pose likelihood. (b) Logarithmic ADC converting I_{LL} into digitized log likelihood.

the model score of the conventional GMM model is better than our HMGM model, the former model is much more complex to implement; meanwhile, each mixture in our HMGM model only requires six transistors.

C. CIM Likelihood Estimator

Fig. 6(a) shows the architecture of a compute-in-memory (CIM) implementation based on the HMGMs for ultralow-power likelihood evaluations. The architecture com-

prises of parallel columns, each constituting a multiinput inverter design in Fig. 4. The gate voltages, V_X, V_Y, and V_Z, are shared among columns. Each column injects a current proportional to HMG to I_{LL} line at the top. The current of all column lines are summed to produce the net current mimicking the likelihood estimation which is then converted to digital domain using an analog-to-digital converter (ADC). Since each column in Fig. 6(a) comprises transistors of identical dimensions, the effect of mixture function weight is accommodated by mapping mixture functions with higher weights on proportionally more number of columns as shown in the figure.

By passing current-mode likelihood estimates from our FG inverter array to a log ADC, logarithm of the likelihood estimates can be performed during digitization itself and an additional digital-domain logarithm operation will not be needed. Therefore, instead of a linear ADC, we have chosen a logarithmic ADC. The architecture of N-bit logarithmic ADC is shown in Fig. 6(b) that converts an analog input voltage (V_{IN}) into digital bits based on the following equation [45]:

$$\log_{10}\left(\frac{V_{\text{IN}}}{V_{\text{range}}}\times 10^{C}\right)=\frac{b_{N-1}2^{N-1}+\cdots+b_0}{2^N}C. \quad (14)$$

Here, C is the code efficiency factor where larger values of C emphasize smaller signals, resulting into higher dynamic range. The logarithmic adaptation of a conventional linear ADC would replace squaring operations with multiplication by 2 and similarly division with subtraction. Therefore, log-ADC is not computationally cumbersome avoiding complex operations like squaring and exponents. To convert analog current, I_{LL}, representing likelihood of depth scan for the pose hypothesis, into analog voltage, we use an Op-Amp with resistive feedback as shown in Fig. 6(b). This also adds voltage bias to the inverter columns. Furthermore, by adjusting the feedback resistance, the same log-ADC can be used for different workloads.

The architecture in Fig. 6 considerably simplifies likelihood evaluations. Since the current of mixture functions is added over a wire, a dedicated adder is not needed. Although the implementation requires data converters: ADC and digital-to-analog converter (DAC), the overheads of ADC/DAC amortize on a larger scale processing architecture comprising many parallel columns. Note that the workload of ADC/DAC does not increase even when the HMG columns in the processing array are increased to operate on higher complexity maps.

D. Programming of the Likelihood Estimator

Transistors in the likelihood estimator array in Fig. 6(a) require VTH (V_{TH}) programming to map the respective HMG function. As we discussed earlier, the voltage for peak current of the inverter can be programmed by Δ along X, Y, and Z input dimensions by programming V_{TH} of the corresponding nMOS and pMOS transistors as $V_{\text{TH},n} \rightarrow V_{\text{TH},n}+\Delta$ and $V_{\text{TH},p} \rightarrow V_{\text{TH},p}-\Delta$. Notably, programming of our likelihood estimator is similar to NAND flashes [46]–[48]. However, our design requires several key modifications since each column combines transistors of both n and p-types, whereas NAND flashes are based on n-type transistors alone.

In Fig. 7(a), to increase V_{TH} of nMOS, we select the respective column by applying a high voltage, VDP, across it. Unselected columns are held at a voltage V_{inhibit}. A programming pulse is applied at the gate of the selected nMOS. A passing gate potential, V_{pass}, is applied to the gate of unselected transistors. V_{inhibit} and V_{pass} are chosen so that the gate to channel potential is not high enough in the unselected transistors to induce a carrier tunneling. These design considerations for V_{inhibit} and V_{pass} in our case are similar to NAND flashes. Under the above programming configurations, electron in the selected nMOS inject to its FG through hot carrier injection (HCI) [49], thereby increasing its V_{TH}. To decrease V_{TH}, the source end of the selected transistor is left floating and high programming pulse is applied at the drain. Due to a higher negative voltage drop between the FG and drain, electrons are detrapped by tunneling to the high-voltage drain terminal.

In Fig. 7(b), pMOS is programmed by applying potential between the gate and n-well body voltage. Each column has separate n-wells and body terminals. Source-drain terminals in the selected column are left floating during programming. V_{TH} of the selected pMOS is decreased by applying programming pulses at the gate of the selected transistor and keeping the body grounded. Due to a high voltage between the gate and the channel, majority carrier electrons tunnel from n-well body to the FG. pMOS rows are unselected by applying a passing gate potential V_{pass}, similar to the case in Fig. 7(a). Inverter columns are unselected by applying a inhibitory potential V_{inhibit} at the body terminal so that the potential between gate and channel is not enough to induce tunneling. V_{TH} of pMOS is increased by applying programming pulses at the body terminal, thereby detrapping electrons from FG to drain terminal.

Notably, pMOS's programmability in the above scheme requires separate n-wells for each column which affects the area efficiency of inverter array. However, this is not a critical concern for most applications since only a few thousand minimum-sized transistors are needed even for complex maps in our approach. Moreover, while our previous discussion considers FG CMOS for mean programmability, the schemes can also be extended to charge trap transistors (CTTs). CTT based on HfO_2-metal gates was recently demonstrated [50], programmable under 2 V. Due to lower voltage operation, with CTT, the programming circuits of inverter array can be simplified to improve overall area efficiency of the array.

VTH programming from TCAD simulations is shown in Fig. 7(c)–(f). Fig. 7(c) shows the TCAD schematic of an FG nMOS, where 3-nm thick poly-silicon FG is stacked between 1-nm thick tunneling oxide (SiO_2) (between the FG and Si substrate) and a 4-nm-thick blocking oxide (HfO_2) between the FG and polysilicon gate on 6-nm thickness. Electron charging of the FG is governed by HCI [49] increasing the VTH of FG-nMOS. FG is discharged by Fowler–Nordheim tunneling [51] reducing the VTH of FG-nMOS. Fig. 7(d) illustrates the charging (programming) and discharging (erasing) at the FG and corresponding changes in the VTH. To program FG-nMOS in Fig. 7(e), we apply a drain pulse of 1 V, the source and substrate terminals are grounded, and VTH programming is shown in the figure for three different

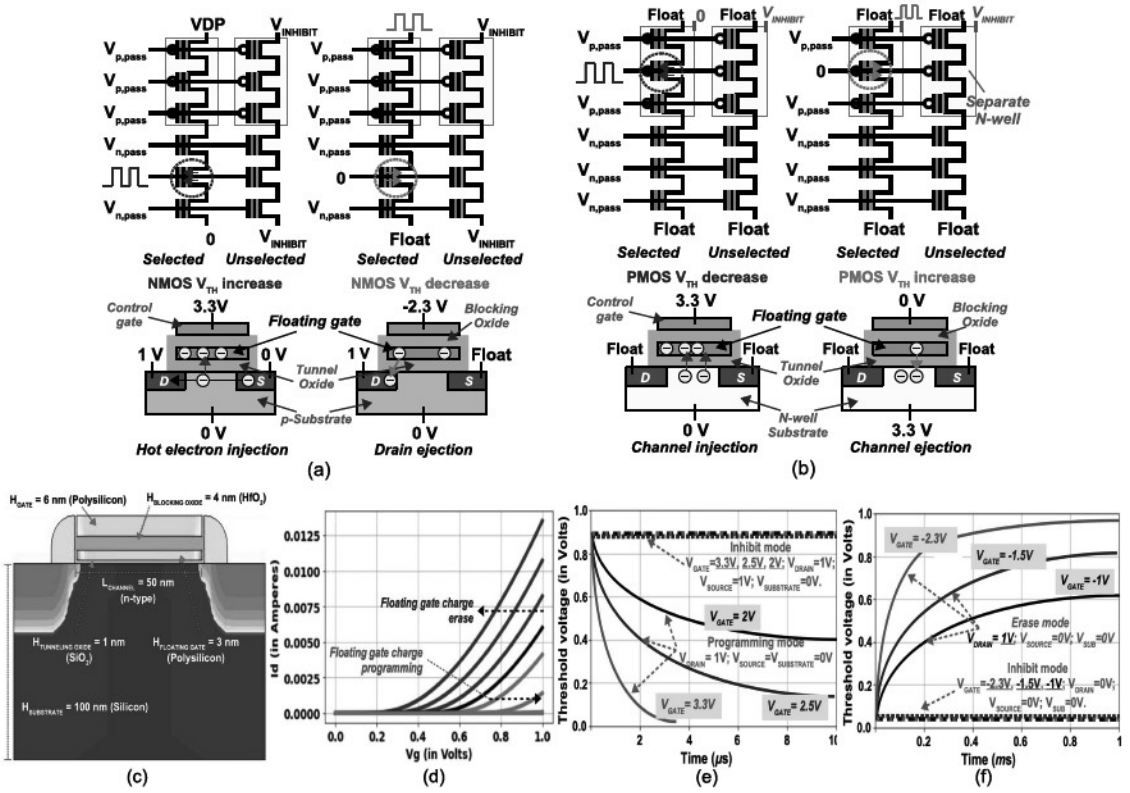

Fig. 7. VTH (V_{TH}) programming in FG inverter columns so as to program the mean of HMGM components. (a) Configuration to increase and decrease nMOS V_{TH}. (b) Configuration to increase and decrease pMOS V_{TH}. (c)–(f) TCAD simulation results for VTH programming in an FG nMOS transistor. (c) TCAD visual of an FG nMOS transistor. (d) FG charging/discharging resulting into VTH shifts. (e) Programming and inhibit configurations for FG-nMOS. (f) Erasing and inhibit configurations for FG-nMOS.

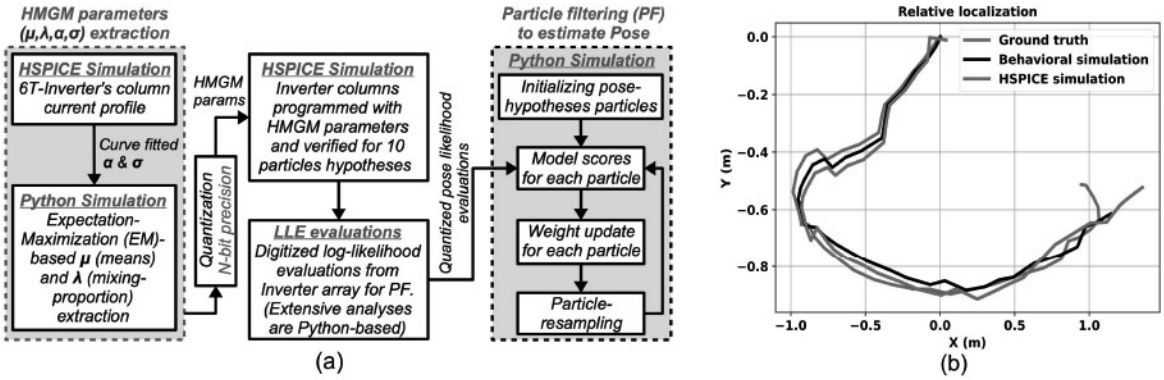

Fig. 8. (a) Methodology for HSPICE and Python-based simulations of the proposed localization framework based on FG inverter array to compute log likelihood of pose during particle filtering (PF). (b) Close correspondence between behavioral and circuit simulation trajectories using 10 pose particles and same random seed.

gate pulses (2, 2.5, and 3.3 V) with duration of the pulses. For the unselected column, programming in inhibited when we short drain and source to 1 V. Similarly, in Fig. 7(f), we illustrate FG-nMOS in erase configuration, where a 1 V pulse is applied at drain–source and substrate terminals are grounded and erasing of FG is shown for three distinct gate pulses (−1, −1.5, and −2.3 V) with the pulse duration. Compared to programming, erasing requires a longer duration pulse at the gate and the drain terminals. Erasing is inhibited for the unselected columns when drain and source terminals are at the same potential. Therefore, the TCAD results show that the FG inverter array can be programmed to intended specifications using row-wise gate programming pulses and column-wise source/drain potential-induced inhibition of unselected columns.

IV. Simulation Methodology

Fig. 8(a) our simulation methodology which integrates HSPICE-based circuit simulations and Python-based functional simulations for power-performance analysis of probabilistic drone localization based on the proposed framework. First, a multiinput 6T inverter in Fig. 4(a) is simulated in HSPICE to extract its HMG switching current against multivariate gate voltage inputs using predictive technology models (PTMs) in 45-nm CMOS process [52]. Extracted characteristics are then curve-fitted based on the proposed

model in (11). From the curve fitting, we get α and σ parameters of the switching current model. Using the switching current model, μ and λ parameters of HMGM-based likelihood function in (12) are learned from Python-based functional simulations of EM procedure in Algorithm 1. Essentially, μ determines the VTH (V_{TH}) that need to be programmed on FG inverter array, and λ determines how many columns of inverter array are used to map the respective mixture function. Since V_{TH} programmability of FG is limited in precision, we map the learned μ parameter as μ_q to FG inverter array with varying degrees of lower precision cases. Our characterization of drone localization on these varying lower precision cases is also discussed in Section VI. Similarly, considering that the inverter array has N columns, the mixture function weight parameter λ also needs to be quantized. To map λ on the inverter array, round($\lambda \times N$) inverter columns are used, therefore, the relation $\lambda_q = \text{round}(\lambda \times N)/N$ determines the corresponding quantized value. Since circuit-level simulations for trajectory extraction are prohibitively expensive—for hundreds of particles and entire depth frames, FG inverter array in Fig. 6(a) needs to be simulated millions of times—we use behavioral simulations using (12). In Fig. 8(a), our behavioral simulation setup is first validated against HSPICE simulations by comparing the prediction for ten particles and with reduced depth frame precision in Fig. 8(b). A very close correspondence between behavioral and circuit simulations can be seen. Even with the ten particles, FG inverter array is simulated thousands of times on widely varying inputs. Therefore, the average power for this case is a good statistical predictor to estimate the power performance characteristics of the entire path trajectory using behavioral simulations. For the trajectory in Fig. 8(b), the output current of inverter array is digitized in log-domain to extract log-likelihood (LL) of various 3-D-projected depth maps. DAC and log-ADC units used in our design are based on prior works [45], [53]. A functionally ideal operation of DAC/ADC is considered. However, since DAC operates on highly capacitive gate rows of inverter array, intermediate analog-mode voltage buffers are considered along with the inverter array. Finally, the remaining steps of particle filtering iteration (model scoring, weight updates, and particle resampling) are simulated functionally again, since our proposed design assumes that these steps are performed in a microcontroller.

V. Characterization on Benchmark Dataset

We use 12 scenes from the RGB-D Scenes Dataset v2 [44] to characterize our proposed localization framework. Each scene in the dataset is a 3-D reconstructed point-cloud formed by aligning a set of video frames. The dataset also contains true camera poses for each frame. The camera pose constitutes the orientation expressed as quaternions (q, w, p, r), and position as (x, y, z) coordinates.

We estimate drone's pose using the particle filtering (PF) approach under both global and relative localization. Under global localization, the drone is completely uncertain about its pose to begin with. By accumulating sequential depth measurements, the drone tracks its movements using PF. Under local localization, the drone is aware of the initial pose. However, since the motion model of drone is stochastic, it needs to track its movements based on depth measurements and a known initial pose as it navigates through the flying space. Under global localization, we initialize our PF setup with hundreds of hypothesized poses (i.e., particles) uniformly distributed throughout the map. All particles (say K in number) are initialized with the same weight $(1/K)$ quantifying that each hypothesis is equally likely. We then project the scanned depth frame to 3-D world coordinates based on a particle j using (5). Corresponding to each particle, a unique 3-D projection of depth scan, $\mathbf{d}^{\mathbf{j}}_{\mathbf{meas}}$, is determined. A model score $\mathcal{MS}(\mathbf{d}^{\mathbf{j}}_{\mathbf{meas}}|\boldsymbol{\Theta})$ for each particle-measurement pair is computed using our inverter array-based likelihood estimator in Fig. 6. By accumulating model score of particles, their weights are updated as

$$\mathbf{w_j} = \frac{\frac{1}{\mathcal{MS}\left(\mathbf{d}^{\mathbf{j}}_{\mathbf{meas}}|\boldsymbol{\Theta}\right)}}{\sum_{j=1}^{K} \frac{1}{\mathcal{MS}\left(\mathbf{d}^{\mathbf{j}}_{\mathbf{meas}}|\boldsymbol{\Theta}\right)}}. \tag{15}$$

To eliminate less likely particles with each measurement, we resample them from the current particle set based on their weights while keeping the total number of particles same as K, similar to [54]. The weight-update followed by resampling results into an updated estimate of drone's pose with uncertainty in the pose illustrated by the variance of resampled particles. We then reassign equal weights to new particles and reiterate all steps from weight-update till resampling for each subsequent depth scan. With this recursive Bayesian estimation, the particles eventually converge resulting into pose prediction with higher confidence. Fig. 9(a) shows such particle filtering in action. A critical goal of this article was to explore mutually suited hardware and learning models. Implementation of GMM-based probabilistic inference models is quite complex for analog computing framework that we have considered here. Although switching current of inverters can emulate the characteristics of a Gaussian current, to represent a GMM model, switching currents from multiple inverters need to be multiplied through analog multipliers which will eclipse simplicity and efficiency of our scheme. Meanwhile, HMGM models can be easily implemented in our analog scheme by stacking inverters in a column and accumulating their switching resistance using Kirchhoff's law. On the other hand, implementation of HMGM-based probabilistic inference is quite cumbersome for digital implementation since it will require divisions and floating-point (FP) operation. Therefore, for a fair comparison between both schemes, we have considered appropriate learning models: HMGM for analog computing and GMM for digital computing.

Fig. 9(b)–(e) shows the trajectory tracking results on Scene-01 based on the approach. Fig. 9(b) shows the point cloud map of Scene-01 along with sample RGB and depth scans from the drone. Fig. 9(c) shows the trajectory tracking under global and relative localization for our approach and the conventional GMM-based approach. Notably, the tracking accuracy of our approach is similar to the conventional approach. Fig. 9(d) shows the confidence in pose estimation corresponding to every input depth frame. Under global

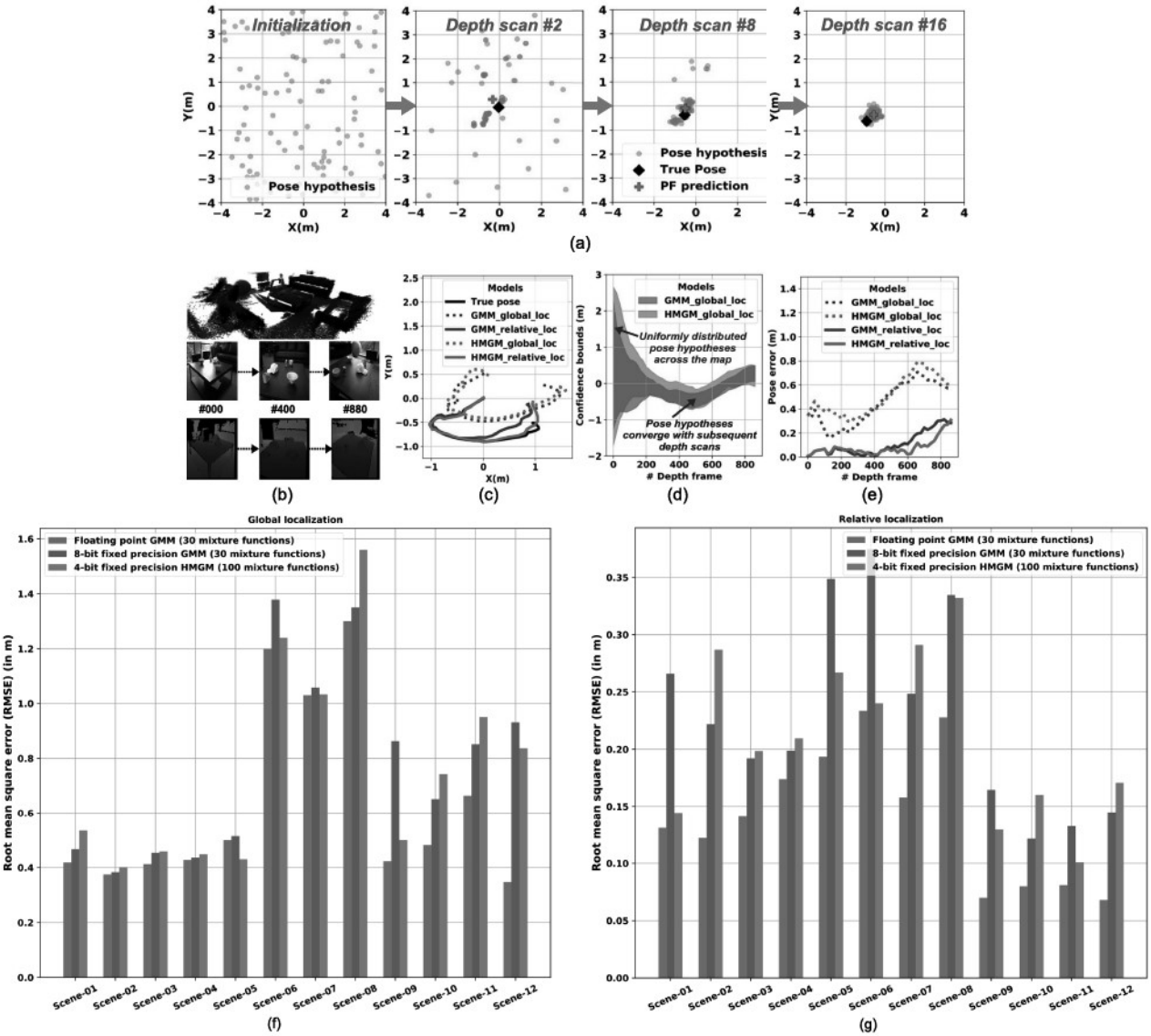

Fig. 9. (a) Particle filtering (PF) on Scene-01 of RGB-D dataset. (b)–(f) Trajectory tracking results on (b) Scene-01. (c) Drone's pose under both global and local (or relative) localization based on both HMGM and GMM. (d) Lower and upper bounds of pose hypotheses with sequential depth scans. (e) Pose error under both global and local localization. (f)–(g) RMSE plots of FP GMM, 8-bit fixed precision GMM and 4-bit fixed precision HMGM-based drone localization in (f) global, and (g) relative (initial position known) settings. The three frameworks are characterized over 12 scenes in the RGB-D dataset.

localization, due to high initial uncertainty, variance bounds on pose particles are high. However, with each increasing measurement, the drone is able to minimize its uncertainty by particle filtering and using the proposed HMGM-based map representation. The bounds converge with subsequent scans illustrating improvement in prediction confidence. The error in position estimates within the scenes for every depth scan is shown in Fig. 9(e) with an upper error limit of ∼0.8 m. The rigorous characterization over multiple scenes is shown in Fig. 9(f)–(g). The localization root mean squared error (RMSE) of an FP GMM-based framework is smaller than that based on fixed-precision GMM for all the scenes. The maximum RMSE of 1.56 m in global localization [Fig. 9(f)] corresponds to HMGM-based framework in scene 8. In relative localization [Fig. 9(g)], the maximum RMSE of 0.37 m corresponds to fixed-precision GMM in scene 6. Note that, the initial pose is known in relative localization unlike in the global localization and hence the pose errors in relative localization are significantly (∼60%) lower. Our proposed HMGM-based localization framework performs close to the existing benchmarks based on FP and fixed precision GMMs.

VI. Impact of Low Precision and Process Variability

We next analyze the implications of quantization on pose tracking of drone. The lower the operating precision, the lower is the area/power requirement which is critical for a drone with limited resource budget. However, excessively low precision can significantly degrade the accuracy of pose prediction. Fig. 10 shows the impact of quantization of DAC, logarithmic-ADC, and VTH (mean) programming on the pose estimation. DAC precision quantizes the 3-D depth projection to inverter array, that is, V_X, V_Y, and V_Z. Log-ADC precision quantizes the digitized log likelihood evaluated by the inverter array. Precision on transistor thresholds quantize the mean of HMGM mixture components and depends on the precision of

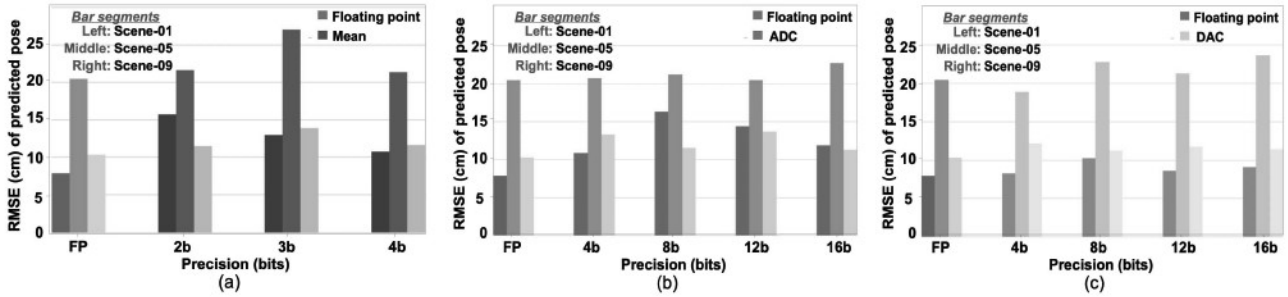

Fig. 10. Implications of low-precision (a) mean, (b) ADC, and (c) DAC on pose prediction (RMSE) characterized over three distinct scenes of RGB-D dataset.

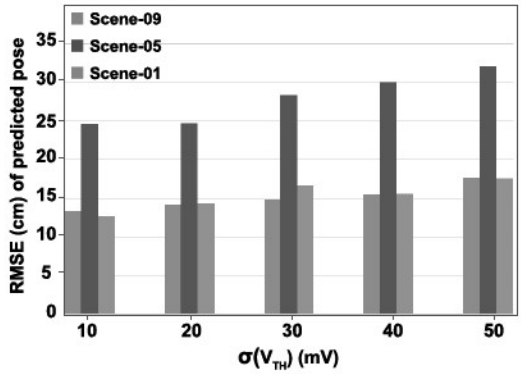

Fig. 11. Process variation analysis on pose prediction at three distinct scenes from RGB-D dataset illustrating RMSE versus variance in VTH variation.

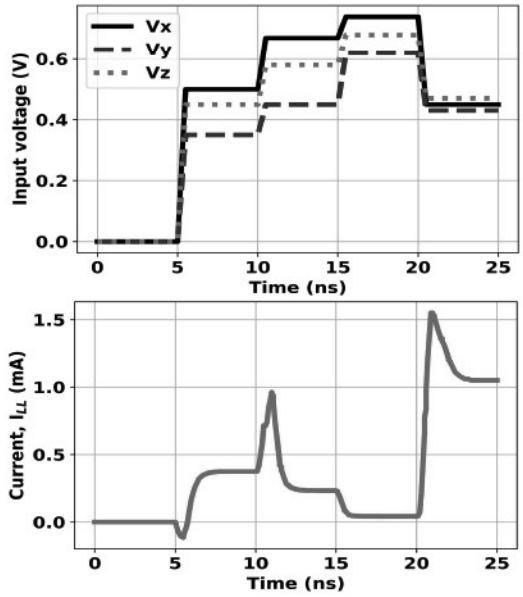

Fig. 12. Transient waveform of analog likelihood computation by an FG inverter array with 500 6T-inverter columns corresponding to a specific choice of multivariate inputs and same VTH across all NFETs and PFETs for the purpose of clear illustration of transitions.

FG memory in the inverter array. Compared to FP precision, the RMSE of predicted poses at lower operating precision is higher as shown in the figure. Our simulation results indicate a nonmonotonic change in RMSE at increasing precision, although a higher precision generally helps in reducing RMSE. Nonetheless, even with very low precision in mean encoding, DAC, and ADC (such as 2, 4, and 4-bit, respectively), the pose estimation accuracy is competitive to the FP precision.

We have also analyzed the impact of process variation on pose prediction in our approach, as shown in Fig. 11. VTH (V_{TH}) variations in the inverter transistors degrade the emulation of 3-D map model using inverter array, thereby resulting in inaccuracy in likelihood evaluation and finally in pose estimation. The impact of V_{TH} variability is analyzed behaviorally using Monte Carlo simulations. V_{TH} of transistors in the inverter array are randomly perturbed from the ideal based on process statistics of V_{TH} variability. This is equivalent to random perturbation of μ_q in the above simulation methodology [Fig. 8(a)]. For hundred random cases, RMSE of the predicted trajectories and ground truth is analyzed. Based on the simulation results, $\sigma(V_{\mathrm{TH}}) < 20$ mV is tolerable in our design. Furthermore, V_{TH}-induced variability of an FG inverter array can be addressed during the programming stage itself. During the programming, a sequence of write-read–verify steps can be performed where the programming pulsewidth can be adjusted if the transistors have innate V_{TH} deviation from the typical and such V_{TH} variability can be corrected by adjusting the programming pulsewidth.

VII. Energy-Performance Characterization

Fig. 12 shows an example transient current (I_{LL}) profile from the inverter array when applying depth pixel projections as step input sequences. In our design, the total time to process one depth pixel projection by the inverter is 5 ns. Considering 100 pose hypotheses (particles) in PF and on average ∼60% nonzero depth pixels in a 640 × 480 resolution depth frame from the RGB-D scenes dataset (note that only nonzero depth pixels are projected and considered for likelihood evaluation), our framework takes ∼19 ms on average to process entire depth frame, achieving the speed of 52 fps (frames per second). The statistics of nonzero depth pixels is based on the considered RGB-D dataset. The likelihood evaluations by our proposed inverter array will not be a bottleneck to the speed. High performance is enabled by parallel processing in an inverter array which processes all components of map representation model simultaneously. Multiple instances of the inverter array would further enhance depth frame processing speed.

The energy consumption of the proposed framework and comparison with digital GMM processor is shown in Fig. 13. Fig. 13(c) shows the power breakdown for the measurement likelihood estimation in our approach. The total energy consumption (with 500 inverter columns emulating 100 mixture components) for the likelihood estimation is 374 fJ

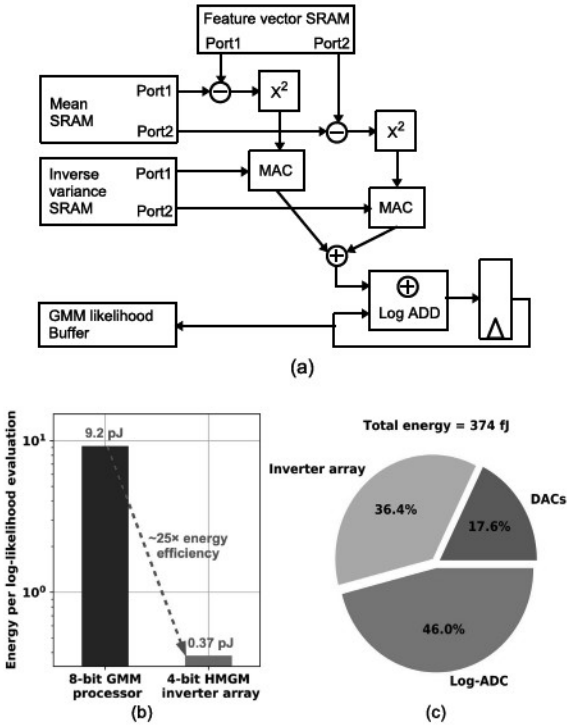

Fig. 13. (a) Digital processor to evaluate GMM-based log-likelihood. (b) Energy comparison of 30 mixture component GMM processor and our proposed 100 mixture component HMGM-based inverter array to compute pose-likelihood. (c) Energy breakdown of proposed inverter array peripherals.

(at 45-nm CMOS process technology). We do not consider energy contribution from the front-end digital processing to project depth pixels to the 3-D space, since the workload for the likelihood estimations is the dominant component (Fig. 3), exceeding by the orders of magnitude than the front-end processing. Log-ADC with several comparators and Op-Amps at multiple pipeline stages consumes 46% of the total energy. The three DACs at the gate inputs contribute to 17.6% and the 500 column inverter array consumes 36.4% of the total energy as shown in the figure. Table I shows the projection of energy of reference designs to our design specifications. The focus of this work is to explore alternative processing schemes for log-likelihood computations using inverter arrays. The respective energy is extracted from HSPICE simulations. The energy consumption of peripheral ADC/DAC is from the referenced designs [40], [48]. Under ideal energy scaling, we have assumed Energy $\propto$ (45 nm/TechNode$_{\text{REF}}$)^2, Energy $\propto$ (1 V/VDD$_{\text{REF}}$)^2, and Energy $\propto$ $2^{(4-\text{Precision}_{\text{REF}})}$, thereby the corresponding energy estimates are extracted. The energy breakdown also indicates that with even more complex map representation model (i.e., the ones requiring more than 100 HMGM mixture functions or more than 500 inverter columns to implement), the total power dissipation in our approach will not increase significantly from the current. To implement a more complex HMGM model, only more inverter columns will be needed and three DACs and one log-ADC will suffice as in the present configuration.

Fig. 13(a) shows a comparative digital pipeline implementing a GMM [39] to evaluate likelihood of depth projections. The design uses dual-port dedicated SRAM modules for mean and inverse variance storage. In each clock cycle, two square and two multiply and accumulate (MAC) units compute exponent terms in a GMM. Exponents of two mixture functions are combined using a log-ADD lookup table to iteratively compute the net log-likelihood. The original design in [39] was implemented in 65-nm CMOS process. Using power consumption estimates of different blocks in datapath from [55] for 45-nm CMOS process, the 8-bit digital GMM processor consumes 9.2 pJ and operating for 30 mixture functions in a 3-D GMM. Comparatively, as shown in Fig. 13(b), our proposed framework's power efficiency is 25$\times$ higher even with 100 mixture functions in HMGM.

TABLE I
ENERGY PROJECTED TO 1 V, 45 nm CMOS, 4-BITS AND 50 MHz

Components	Power (reference)	Energy (proposed design)
Log-ADC	2.54mW @ 1.62 V, 0.18 µm, 8-bits, 22 MS/s [40]	172 fJ
DAC	27mW @ 1.2 V, 0.13 µm, 10-bits, 1.6 GHz [48]	66 fJ (for 3 DACs)
INV columns	-	130 fJ

VIII. CONCLUSION

Energy-efficient, low-latency, on-board processing of self-navigation algorithms is imperative in insect-scale drones. Our proposed FG CIM (FG-CIM) framework performs confidence-aware localization of the drone with ultralow-power consumption. The framework uniquely represents the domain map with the 3-D HMGM density model, characteristic of the inverter columns that construct FG-CIM. The scalable design processes depth frames for a large number of pose hypotheses at very high speed due to massive parallelism, paving the way for robust, ultralow-power spatial intelligence in insect-sized drones.

ACKNOWLEDGMENT

The authors acknowledge insightful discussions with Wilfred Gomes and Amir Khosrowshahi that helped shape ideas for this work.

REFERENCES

[1] Y. M. Chukewad, J. James, A. Singh, and S. Fuller, "RoboFly: An insect-sized robot with simplified fabrication that is capable of flight, ground, and water surface locomotion," 2020, *arXiv:2001.02320*. [Online]. Available: http://arxiv.org/abs/2001.02320

[2] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," *Nature*, vol. 570, no. 7762, pp. 491–495, Jun. 2019.

[3] A.-K. Mahlein, "Plant disease detection by imaging sensors—Parallels and specific demands for precision agriculture and plant phenotyping," *Plant Disease*, vol. 100, no. 2, pp. 241–251, Feb. 2016.

[4] N. Chebrolu, P. Lottes, T. Labe, and C. Stachniss, "Robot localization based on aerial images for precision agriculture tasks in crop fields," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1787–1793.

[5] A. Kroll, W. Baetz, and D. Peretzki, "On autonomous detection of pressured air and gas leaks using passive IR-thermography for mobile robot application," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 921–926.

[6] C. Heyer, "Human–robot interaction and future industrial robotics applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 4749–4754.

[7] L. Apvrille, T. Tanzi, and J.-L. Dugelay, "Autonomous drones for assisting rescue services within the context of natural disasters," in *Proc. 31th URSI Gen. Assem. Sci. Symp. (URSI GASS)*, 2014, pp. 1–4.

[8] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "LSAR: Multi-UAV collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55817–55832, 2019.

[9] V. K. KV, B. B. Khot, E. Oh, and B. Swain, "Home, office security, surveillance system using micro mobile drones and IP cameras," U.S. Patent 9819911, Nov. 14, 2017.

[10] L. M. Belmonte, R. Morales, A. S. García, E. Segura, P. Novais, and A. Fernández-Caballero, "Assisting dependent people at home through autonomous unmanned aerial vehicles," in *Proc. Int. Symp. Ambient Intell.* Springer, 2019, pp. 216–223.

[11] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood, "Controlled flight of a biologically inspired, insect-scale robot," *Science*, vol. 340, no. 6132, pp. 603–607, 2013.

[12] J. James, V. Iyer, Y. Chukewad, S. Gollakota, and S. B. Fuller, "Liftoff of a 190 mg laser-powered aerial vehicle: The lightest wireless robot to fly," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–8.

[13] S. B. Fuller, "Four wings: An insect-sized aerial robot with steering ability and payload capacity for autonomy," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 570–577, Apr. 2019.

[14] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A fully integrated energy-efficient visual-inertial odometry accelerator for autonomous navigation of nano drones," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2018, pp. 133–134.

[15] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, no. 7553, pp. 460–466, 2015.

[16] E. F. Helbling and R. J. Wood, "A review of propulsion, power, and control architectures for insect-scale flapping-wing vehicles," *Appl. Mech. Rev.*, vol. 70, no. 1, pp. 010801-1–010801-9, Jan. 2018.

[17] J.-H. Yoon and A. Raychowdhury, "NeuroSLAM: A 65-nm 7.25-to-8.79-TOPS/W mixed-signal oscillator-based SLAM accelerator for edge robotics," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 66–78, Jan. 2021.

[18] *How Much Power is Needed to Hover?* Accessed: Oct. 28, 2015. [Online]. Available: https://justdrones.com.au/how-much-power-is-needed-to-hover/

[19] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Uavs for smart cities: Opportunities and challenges," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2014, pp. 267–273.

[20] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118. Springer, 2012.

[21] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, May 2015.

[22] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[23] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 4762–4769.

[24] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE Pervasive Comput.*, vol. 2, no. 3, pp. 24–33, Jul./Sep. 2003.

[25] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Sep. 1999, pp. 1322–1328.

[26] S. Thrun, "Particle filters in robotics," in *Proc. 18th Conf. Uncertainty Artif. Intell.* San Mateo, CA, USA: Morgan Kaufmann, 2002, pp. 511–518.

[27] R. W. Wolcott and R. M. Eustice, "Fast LIDAR localization using multiresolution Gaussian mixture maps," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2814–2821.

[28] A. Dhawale, K. S. Shankar, and N. Michael, "Fast Monte-Carlo localization on aerial vehicles using approximate continuous belief representations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5851–5859.

[29] H. Huang, H. Ye, Y. Sun, and M. Liu, "GMMLoc: Structure consistent visual localization with Gaussian mixture models," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5043–5050, Oct. 2020.

[30] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Feb. 2012.

[31] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1–4.

[32] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 6101–6108.

[33] N. Iliev, A. Gianelli, and A. R. Trivedi, "Low power speaker identification by integrated clustering and Gaussian mixture model scoring," *IEEE Embedded Syst. Lett.*, vol. 12, no. 1, pp. 9–12, Mar. 2020.

[34] A. Gianelli, N. Iliev, S. Nasrin, M. Graziano, and A. R. Trivedi, "Low power speaker identification using look up-free Gaussian mixture model in CMOS," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL CHIPS)*, Apr. 2019, pp. 1–3.

[35] P. Shukla, A. Shylendra, T. Tulabandhula, and A. R. Trivedi, "MC2RAM: Markov chain Monte Carlo sampling in SRAM for fast Bayesian inference," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[36] F. Masuoka, M. Asano, H. Iwahashi, T. Komuro, and S. Tanaka, "A new flash E^2PROM cell using triple polysilicon technology," in *IEDM Tech. Dig.*, Dec. 1984, pp. 464–467.

[37] D. Schinke, N. Di Spigna, M. Shiveshwarkar, and P. Franzon, "Computing with novel floating-gate devices," *Computer*, vol. 44, no. 2, pp. 29–36, Feb. 2011.

[38] A. Basu *et al.*, "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[39] M. Price, J. Glass, and A. P. Chandrakasan, "A 6 mW, 5,000-word real-time speech recognizer using WFST models," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 102–112, Jan. 2015.

[40] A. Shylendra, S. H. Alizad, P. Shukla, and A. R. Trivedi, "Non-parametric statistical density function synthesizer and Monte Carlo sampler in CMOS," in *Proc. 33rd Int. Conf. VLSI Design 19th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2020, pp. 19–24.

[41] A. Shylendra, P. Shukla, S. Mukhopadhyay, S. Bhunia, and A. R. Trivedi, "Low power unsupervised anomaly detection by non-parametric modeling of sensor statistics," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 8, pp. 1833–1843, Aug. 2020.

[42] A. R. Trivedi and A. Shylendra, "Ultralow power acoustic feature-scoring using Gaussian I-V transistors," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[43] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[44] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 3050–3057.

[45] J. Lee *et al.*, "A 2.5 mW 80 dB DR 36 dB SNDR 22 MS/s logarithmic pipeline ADC," *IEEE J. Solid-State Circuits*, vol. 44, no. 10, pp. 2755–2765, Oct. 2009.

[46] S. Aritome, *NAND Flash Memory Technologies*. Hoboken, NJ, USA: Wiley, 2015.

[47] Y. J. Park *et al.*, "3-D stacked synapse array based on charge-trap flash memory for implementation of deep neural networks," *IEEE Trans. Electron Devices*, vol. 66, no. 1, pp. 420–427, Jan. 2019.

[48] S. Lee *et al.*, "A 1 Tb 4b/cell 64-stacked-WL 3D NAND flash memory with 12 MB/s program throughput," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 340–342.

[49] E. Takeda, Y. Nakagome, H. Kume, and S. Asai, "New hot-carrier injection and device degradation in submicron MOSFETs," *IEE Proc. I-Solid-State Electron Devices*, vol. 130, no. 3, pp. 144–150, Jun. 1983.

[50] F. Khan, E. Cartier, J. C. S. Woo, and S. S. Iyer, "Charge trap transistor (CTT): An embedded fully logic-compatible multiple-time programmable non-volatile memory element for high-k-metal-gate CMOS technologies," *IEEE Electron Device Lett.*, vol. 38, no. 1, pp. 44–47, Jan. 2017.

[51] M. Lenzlinger and E. Snow, "Fowler-Nordheim tunneling into thermally grown SiO_2," *J. Appl. Phys.*, vol. 40, no. 1, pp. 278–283, 1969.

[52] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Trans. Electron Devices*, vol. 53, no. 11, pp. 2816–2823, Nov. 2006.

[53] P. Palmers and M. J. Steyaert, "A 10–bit 1.6-GS/s 27-mW current-steering D/A converter with 550-MHz 54-dB SFDR bandwidth in 130-nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 11, pp. 2870–2879, Nov. 2010.

[54] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *Proc. IEEE Nonlinear Stat. Signal Process. Workshop*, Sep. 2006, pp. 79–82.

[55] Q. Xie, X. Lin, Y. Wang, S. Chen, M. J. Dousti, and M. Pedram, "Performance comparisons between 7-nm FinFET and conventional bulk CMOS standard cell libraries," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 8, pp. 761–765, Aug. 2015.