On the Compute Cost of Autonomy at the Edge

Sertac Karaman & Vivienne Sze

1 Introduction

As modern vehicles increasingly rely on machine learning (ML) to enhance safety and convenience, the computational demands of these systems are rapidly rising. Advanced Driver Assistance Systems (ADAS), now common in many production cars, depend on onboard ML models to interpret sensor data and support functions such as lane keeping, adaptive cruise control, and automated emergency braking. Leading-edge vehicles, including those marketed as semi-autonomous, go further—processing high-resolution sensor streams in real-time to enable features such as automated lane changes and limited self-driving under human supervision. These capabilities require substantial onboard computing infrastructure. For instance, Tesla vehicles are known to carry computers that draw over 70 watts of continuous power, dedicated primarily to autonomy-related tasks.

Looking ahead, the trajectory toward fully autonomous vehicles will likely demand significantly more compute power. As perception systems grow more complex, and as vehicles take on the full burden of driving in diverse environments without human oversight, the models required to ensure safety, redundancy, and reliability will become even more computationally intensive. Industry estimates and experimental prototypes suggest that onboard computing demands may reach or exceed 1 kilowatt in fully autonomous platforms—an order of magnitude increase over today's deployments.

This trend raises critical questions about the energy sustainability of future mobility. Unlike traditional vehicle systems, which primarily draw power for mechanical operations, autonomous driving introduces a new and growing source of energy consumption: real-time, high-performance computation. As this footprint scales across millions of vehicles, the aggregate impact on transportation sector emissions may be substantial. Understanding and mitigating this energy and carbon cost is essential for ensuring that future mobility solutions remain both technologically advanced and environmentally responsible.

In this report, we outline the results of the research conducted under the Mobility Initiative program at the Massachusetts Institute of Technology (MIT). The research focuses on the energy and compute cost of autonomy at the edge, specifically in the context of autonomous vehicles. We explore the implications of increasing computational demands on energy consumption and carbon emissions, and we propose strategies for optimizing onboard compute systems to minimize their environmental impact. Our research has lead us to examine two key drivers:

- The mapping component of autonomous driving systems, which is critical for real-time navigation and obstacle avoidance.
- The artificial-neural-network (ANN) based perception systems, which are essential for interpreting sensor data and making driving decisions.

We find that these are the most energy-intensive components of autonomous driving systems, and we propose several strategies for optimizing their performance.

2 Mapping without HD Maps

A key driver of compute costs in autonomous driving systems is the mapping component, which is responsible for creating and maintaining a detailed representation of the vehicle's environment. Traditional mapping approaches rely on high-definition (HD) maps, which are pre-computed and stored in the vehicle's memory. However, these maps can be large and require significant computational resources to update and maintain.

We envision future chips that utilize simpler maps that occupy less memory and are easier to update. These maps would be constructed in real-time using sensor data, allowing the vehicle to adapt to changing environments without relying on pre-computed HD maps. This approach has the potential to significantly reduce the computational burden associated with mapping, while also improving the vehicle's ability to navigate dynamic environments. During the execution of algorithms, the energy consumption of memory operations (*e.g.*, reading and writing data stored in cache and DRAM) could dominate the total compute energy. For instance, the energy required for accessing on-chip memory (*e.g.*, cache) is more than an order-of-magnitude higher than that when performing a 32-bit multiplication [24]. The energy consumption of memory access increases with the size and distance of the memory from the processor. Within the same chip, accessing a higher-level L2 cache (a few MBs) requires up to an order-of-magnitude more energy than lower-level L0 and L1 caches (a few KBs). However, accessing data stored in a larger, off-chip memory such as DRAM (GBs of storage) requires more than two orders-of-magnitude higher energy than smaller, on-chip (local) CPU caches [24]. The memory (capacity) usage of an algorithm not only consists of output variables but also input and temporary variables allocated during computation. Thus, algorithms designed for energy-constrained devices should be *memory efficient* such that: *i)* the number of *memory accesses* do not dominate; *ii)* amount of *memory (capacity) overhead* for storing input and temporary variables is small enough to remain in lower-level caches.

Many of the above-mentioned applications requires the long-term interaction between the user / device with its immediate environment. For instance, VR headsets need to inform the user when they are about to collide into objects from the physical world. Micro-robots need to avoid obstacles during space exploration. To ensure the safety of these interactions, devices need to maintain an accurate 3D map of the environment that is not only compact enough for on-device storage but also efficiently constructed from sensory data in real time under severe power constraints. However, existing algorithms [23, 41] are not suitable because they require a large memory overhead for storing

temporary variables during map construction.

In this project, we propose memory-efficient algorithms that efficiently construct 3D maps using Gaussians from different sensor modalities such as depth and RGB cameras. We not only demonstrate a system that enable real-time 3D map construction on energy-constrained devices but also emphasize the necessity of developing memory-efficient algorithms for enabling other applications on these devices.

2.1 Related Work

Constructing an accurate and compact representation of the 3D environment is crucial for ensuring safety during the interaction between energy-constrained devices and their users with their environment. During the past decades, different types of maps are used for different downstream applications. In this section, we review related works on occupancy and photo-realistic maps and their associated specialized hardware.

Occupancy Maps from Depth Camera For robotic applications such as path planning and autonomous exploration, constructing an accurate representation for only obstacles in the environment is not sufficient. In fact, each region in the environment needs to be classified into one of the following three states: occupied (where an obstacle exists), free (where no obstacle exists), and unexplored (where the robot has not visited yet). These states can be elegantly captures by the probabilistic modeling of *occupancy* (*i.e.*, whether or not an obstacle exists) at every location in the 3D environment such that occupied region has an occupancy of one, free region has an occupancy of zero, and unexplore region has an occupancy of 0.5. A distribution that captures how occupancy varies across 3D space is known as an *occupancy map*, which is typically constructed with depth measurements (e.g., from depth camera or LIDAR) and groundtruth poses.

Many frameworks proposed different models to represent the distribution of the occupancy probability (*i.e.*, the likelihood that a region contains an obstacle) across the 3D environment. These models exhibit different trade-offs in memory and computational efficiency during the construction and querying of the map. Some of the most popular mapping frameworks discretize the environment into cubic regions (*i.e.*, grids in 2D and voxels in 3D) such that each region contains a Bernoulli random variable representing the occupancy probability and is assumed to be spatially independent of each other. One of the earliest 2D mapping frameworks, the occupancy grid map [14], discretizes the environments into equally-sized grids. However, the map size is prohibitively large in 3D because the size scales cubically with the dimensions of the voxels and the environment.

To relax the spatial independence assumption in discrete map representations, Gaussian Process (GP) was proposed to estimate a continuous distribution of occupancy [47] using a covariance function that captures the spatial correlation among all sensor measurements. Since GP requires the storage of *all* sensor measurements (since the beginning of the robotics experiment) to update the covariance function, the memory overhead scales with the total number of measurements N. During a map query, the covariance function generates a large matrix that requires $O(N^3)$ to invert, which greatly reduces the query efficiency.

To create an extremely compact representation of the environment, several frameworks compress the sensor measurements using a set of parametric functions (*e.g.*, Gaussians or other kernels) which are then used to infer occupancy. One of the well-known semi-parametric representations is the Normal Distribution Transform Occupancy Map (NDT-OM) [51] that partitions the environment into large voxels such that measurements within each voxel are represented by a Gaussian. Since measurements within a voxel could belong to multiple objects, representing them with a single Gaussian often leads to a loss of accuracy in the resulting map. To further reduce map size, recent frameworks, such as Hilbert Map (HM) [20], Fast Bayesian Hilbert Map (Fast-BHM) [61], Variable Resolution GMM (VRGMM) map [48], Hierarchical GMM (HGMM) map [53], compress sensor rays into special kernels (in HM) or Gaussians (in VRGMM and HGMM). Such compression is performed using techniques such as Quick-Means (QM) [20], Hierarchical Expectation-Maximization (H-EM) [12], Region Growing (RG) [10], Self-Organizing GMMs (SOGMM) [19], and Integrated Hierarchical GMMs (IH-GMM) [18]. However, these techniques require significant memory overhead to store all sensor measurements (more than 300,000 pixels in a 640×480 depth image) due to their *multi-pass* processing.

Photo-realistic Maps from Monocular RGB Camera To relax the requirement for both depth images and ground truth poses, prior frameworks are proposed simultaneously localize and construct a map of the environment (i.e., Simultaneous Localization and Mapping or SLAM) using a monocular RGB camera. These frameworks differ in the techniques that are used for localization and geometric primitives that are used for map construction. For instance, traditional SLAM frameworks [7, 15, 9, 42] extract a set of features (such as corners) that are common across images for both localization and mapping. Although these frameworks are often memory-efficient and real-time, the amount of unique features is very sparse which produces a map with very low coverage of the environment.

To provide a photo-realistic reconstruction of the environment, neural-based frameworks, such as GO-SLAM [60], NICER-SLAM [62], and iMODE [39] represent the environment using a Neural Radiance Field (NeRF). Due to the volumetric rendering required for training NeRFs, the training process is computationally intensive. Thus, most of these frameworks propose techniques that accelerate training, some of which include i) using a hybrid scene representation with the voxel grids (in NICER-SLAM) or hash table (in GO-SLAM), and ii) training on a carefully selected subset of input images (in most prior works including iMODE).

Even though throughput was enhanced by these techniques, almost all Neural SLAM frameworks suffer from *catastrophic forgetting*, which is reduced by periodic re-training on images acquired throughout the entire experiment. Thus, these images need to be stored as overhead in memory, which quickly grows with the duration of the experiment to dominate the total memory usage.

To improve throughput and achieve photo-realistic rendering, recent frameworks, such as MonoGS [41], Photo-SLAM [25], and SplatSLAM [28], use Gaussian Splatting

(GS) to train learnable Gaussians for 3D representation. These frameworks propose different localization techniques to complement GS. For instance, both MonoGS and SplatSLAM localize the camera against the global map via minimizing a photometric cost function, while Photo-SLAM utilizes ORB-SLAM [7]. Similar to Neural SLAM, current Gaussian SLAM frameworks also suffer from *catastrophic forgetting* and thus require the storage of a large number of images to periodically retrain all Gaussians.

2.2 Results

In this section, we present two contributions that enable memory-efficient 3D map construction using Gaussians on energy-constrained devices. In Section 2.2, we propose GMMap, a continuous occupancy map that is not only compact to store but also efficiently constructed from depth images and groundtruth pose with up to 88% less memory overhead compared with prior works. On devices that lack the depth camera, we propose GEVO in Section 2.2 that enables memory-efficient construction of a photo-realistic map from only a monocular RGB camera with up to 94× lower memory overhead than prior works.

GMMap: Memory-Efficient Continuous Occupancy Map Using Gaussian Mixture Model In this project, we proposed a continuous occupancy map using Gaussian mixture model (GMM), called GMMap, that is efficiently constructed from depth images and poses. To significantly reduce memory overhead compared with prior works, GMMap compresses each depth image into a compact local GMMap G_t using SPGF* (an extension from our SPGF algorithm [35]) which processes each image row-by-row in a single pass. Thus, only a single pixel from the image is required in memory at any time. Since the level sets of Gaussians are ellipsoids, the local GMMap G_t is visualized as red (representing obstacles) or blue (representing obstacle-free regions) ellipsoids in Figure 1. Unlike prior multi-pass approaches [20, 53, 12, 48, 10, 19], SPGF* exploits the connectivity of surface geometries embedded in each depth image to achieve highly accurate Gaussian construction while avoiding the storage of the entire image in memory.

After a local GMMap G_t is created, it is used to incrementally update the global map M_{t-1} by fusing overlapping Gaussians that represent the same region in the 3D environment, as illustrated in Figure 2. In prior works [23, 11, 51, 53], the ray associated with each pixel in the image is cast into the global map to determine where such overlap occurs. Since these rays (more than 300,000 pixels in a 640×480 image) emanate outwards from the sensor origin, accessing the global map in memory along these rays often lacks spatial / temporal locality for effective cache usage and leads to higher number of DRAM accesses. Since these rays are compactly represented by Gaussians in GMMap, using a R-tree (i.e., a tree of bounding boxes [21] that enclose Gaussians) to determine where overlap occurs greatly reduces the number of memory accesses.

Using a low-power ARM Cortex A57 CPU, GMMap can be constructed in real-time at up to 60 images per second. Compared with prior works, GMMap maintains high

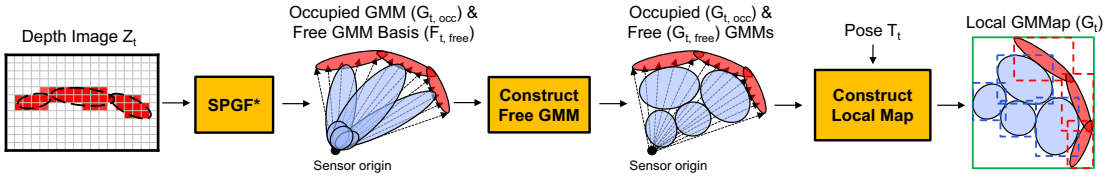


Figure 1: **Per-image GMM construction**: Constructing a local GMMap G_t that accurately represents both occupied and free regions from the current depth image Z_t obtained at pose T_t. Rays associated with each pixel in the depth image are illustrated with dotted arrows. Occupied and free GMMs are illustrated with red and blue ellipsoids, respectively. Dotted rectangles in the map G_t represent the bounding boxes at the leaf nodes of the R-tree. The green rectangle represents the bounding box at the root node of the R-tree that encloses the entire map G_t.

accuracy while reducing the map size by at least 56%, memory overhead by at least 88%, DRAM access by at least 78%, and energy consumption by at least 69%. Thus, GMMap enables real-time 3D mapping on energy-constrained robots.

GEVO: Memory-Efficient Monocular Visual Odometry Using Gaussians To enable the efficient construction of a photo-realistic map using a monocular RGB camera, we propose a memory-efficient framework called GEVO. Unlike depth images, each RGB image does not explicitly encode any geometric information about objects in the 3D environment. Thus, the geometry of the 3D environment needs to be inferred using an optimization process guided by RGB images captured from diverse viewpoints. To achieve real-time operation, many existing frameworks [7, 15, 9, 42] optimize the pose and map using a small sliding window of images (i.e., typically 8 - 10 images). However, the map tends to catastrophically forget and degrade over time after the sliding window has passed (see Figure 3a vs. 3b). To alleviate forgetting, prior frameworks *additionally* store a large number of past images outside the current sliding window to repeatedly retrain the map. Unfortunately, the overhead memory used to store these images dominates by occupying up to 95 % of the total memory and is orders of magnitude higher than both the current sliding window and the map itself.

In GEVO, we significantly reduce memory overhead by *rendering* past images from the existing map instead of storing them in memory. However, without employing additional techniques, the fidelity of these images tends to slowly degrade over time due to the artifacts in the map caused by forgetting, which results in noisy guidance to the optimization process. Thus, using these images to guide Gaussian optimization via splatting (i.e., GS [30]) alone is insufficient for constructing a high-fidelity map. To complement GS, GEVO contains the following procedures to further reduce incorrect occlusion and overfitting due to catastrophic forgetting:

1. **Occupancy-Preserving Initialization**: To reduce incorrect occlusions, Gaussians that lie within the obstacle-free regions are pruned. Thus, in addition to

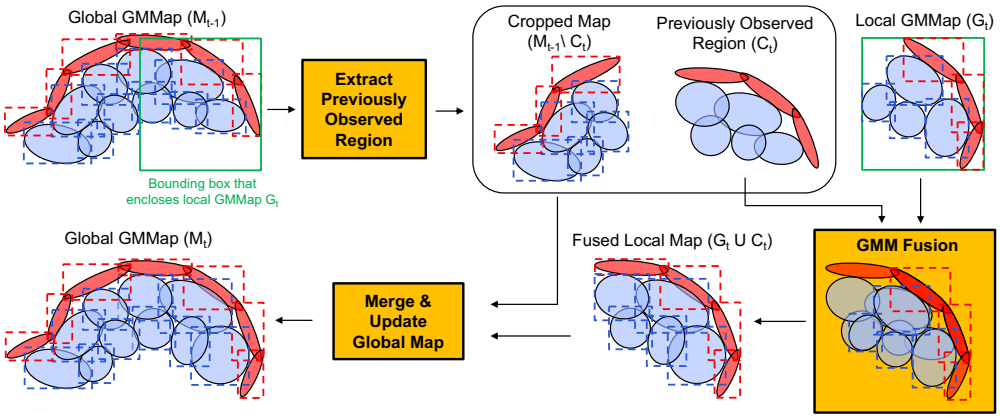


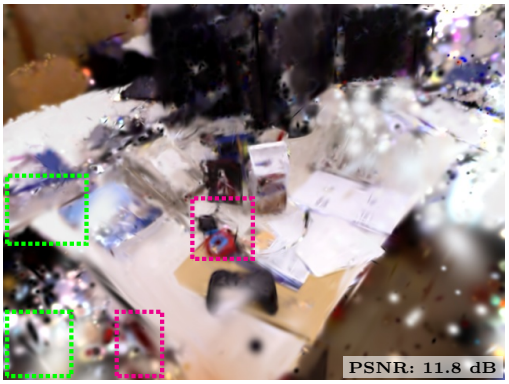
Figure 2: **Globally-consistent GMM fusion**: Constructing the current global GMMap M_t by fusing the local GMMap G_t into the previous global GMMap M_{t-1}. The bounding box (green rectangle) of local map G_t is used to determine the Gaussians C_t in the global map M_{t-1} that overlaps with G_t . Occupied and free GMMs are illustrated with red and blue ellipsoids, respectively. Dotted rectangles represent the bounding boxes at the leaf nodes of the R-tree.

representing obstacles, Gaussians representing free regions are initialized to identify incorrect occlusions.

2. **Consistency-Aware Optimization:** To reduce overfitting of the map to the current window, we only optimize a small subset of Gaussians that are both inconsistent and sufficiently visible to the camera. To ensure rendered images maintain high fidelity, we locally optimize noisy Gaussians created from the current sliding window before merging them to the map for global optimization.

Across a variety of environments, GEVO achieves comparable map fidelity (see Figure 3c) and reduces the memory overhead to around 58MBs, which is up to 94× lower than prior works. Thus, GEVO makes a significant stride towards the deployment of GS-based SLAM on low-energy devices.

(a) Reconstruction before catastrophic forgetting



(b) MonoGS [41] no past images
8 images stored (7 MB)

(c) GEVO (This work)
8 images stored (7 MB)

(d) MonoGS [41]
114 images stored (100 MB)

Figure 3: During online GS-based SLAM, the map (consisting of 3D Gaussians) is built by rendering and optimizing at each viewpoint using a sliding window buffer of images. a) The region visible during the current sliding window achieves high fidelity after initial optimization. b) However, without storing and retraining the map on a large number of past images, the fidelity of the same region degrades over time due to forgetting (artifacts in rectangles). c) While alleviating forgetting, our GEVO avoids storing past images to reduce the memory overhead. d) To achieve similar map fidelity, MonoGS [40] stores all past keyframes and incurs a memory overhead of at least 50× higher than the size of the map.

3 Adaptive Neural Networks for Perception

A second key driver of compute costs in autonomous driving systems is the artificial-neural-network (ANN) based perception systems, which are essential for interpreting sensor data and making driving decisions. In this section, we focus on the task of depth estimation from a camera image. This task is particularly important for those cars that do not include depth sensors such as LIDARs. We examine this task in detail in the coming sections, describing the current state of the art. We then propose a new way to train for this task on the fly. This will allow us to adopt a smaller model for the task, which will reduce the compute cost of the system. The model will be trained while the car is driving. Even though there is some added on-the-fly training cost, the overall compute cost of the system will be reduced when we consider the inference costs as well. This system performs best if the car wil be utilized in the same environment for long periods of time, when retaining costs will be minimal.

3.1 Related Work

Monocular depth estimation has been an active field of research for several years. Eigen et al. used a convolutional neural network (CNN) to predict depth from a single image [13] and started a new field of research for the past decade into DNN-based monocular depth estimation [37, 32, 57, 49, 50, 4, 59, 43, 45, 1] compared to older works using hand-crafted features [52]. Ranftl et al. showed improvement in accuracy for relative depth by using a transformer based architecture for the encoder [50], which many of the state-of-the-art monocular depth estimation methods use for the backbone [4, 59, 43, 45, 1].

For learning-based approaches, two kinds of uncertainty can contribute to errors in predictions: (1) *aleatoric* or data uncertainty that quantifies uncertainty inherent to the data that cannot be reduced (*e.g.,* dark, blurry images) and (2) *epistemic* or model uncertainty that quantifies uncertainty in the model weights (*e.g.,* seeing an object not previously seen in training distribution) [29]. While more training can reduce epistemic uncertainty, it does not reduce aleatoric uncertainty. While aleatoric uncertainty can be computed relatively cheaply via a modified negative log-likelihood loss function [44] [29], computing epistemic uncertainty is computationally expensive.

To compute epistemic uncertainty, most methods involve assembling diverse opinions from multiple models and measuring the disagreement. The state-of-the-art approach is using an ensemble of M networks that each predict a mean depth and aleatoric variance. The ensemble members' predictions are combined via a new Gaussian whose mean and variance is parameterized by the mean and variance of a mixture of Gaussians from each ensemble member's prediction [33, 46]. Another active avenue of research is Bayesian neural networks (BNNs) where each weight is given by a nonparametric distribution and multiple inferences are run with different samples from the weight distributions; however, while BNNs have theoretical guarantees, they are very expensive to train, requiring Monte Carlo Markov Chain (MCMC) approaches for the weight distributions, leading to limited use cases of simple classification tasks [27, 22]. Variational inference

based approaches where each weight distribution is assumed to have some parameteric form (*e.g.,* Gaussian, Bernouilli) is also a popular technique, though it has not yet outperformed ensembles [6, 27, 17]. Monte Carlo Dropout (MC-Dropout) is an extremely popular technique due to its ease of use that can be understood as variational inference with Bernoulli weights [17]; at test time, p percent of weights are randomly dropped out over M inferences, and the variance of the depth estimates is the epistemic uncertainty. However, MC-Dropout is known to produce overconfident predictions [46].

Since requiring multiple inferences per image can be cost-prohibitive, a recent avenue of research in the community is to make epistemic uncertainty more efficient. Single-pass methods such as evidential learning [2] and prior networks [38] require only one inference per image. These methods depend on learned epistemic uncertainty and do not provide guarantees on uncertainty quality. In addition, prior networks require seeing "out-of-distribution" (OOD) examples during training which is a strict requirement not often met.

3.2 Uncertainty from Motion (UfM)

State-of-the-art methods for epistemic uncertainty such as ensembles or MC-Dropout require M inferences per image, either from running M different models in the ensemble approach or M different samples in BNN-based approaches. There exists a gap in deploying uncertainty estimation on resource-constrained applications since running M inferences per image is too computationally expensive.

In this work, we propose an algorithm called Uncertainty from Motion (UfM) that exploits redundancy in multiple views such that only one inference per image has to be run while maintaining ensemble uncertainty quality; this work is published at ICRA 2022 [55]. The key idea behind UfM is that we can improve efficiency of epistemic uncertainty estimation by recognizing that since robots operate on video inputs, points in 3D space are seen over multiple views. Rather than asking an ensemble of DNNs to make predictions on a single image to measure their variance, we can cycle through running one ensemble member on each image and calculate the variance across multiple views of the same point in 3D space. In order for the estimation to be dense and still lightweight, a key insight behind UfM is that we can use noisy correspondences between images in order to estimate which pixels between neighboring images are repeated views of the same point in 3D space.

Specifically, when UfM is applied to ensembles (referred to as ensemble-UfM), given an ensemble $\theta_{1:M}$ of M ensemble members, we cycle through the M networks on the images in the sequence by running only a single ensemble member θ_m on the tth image where $m = t$ modulo M. From running θ_m on the current image X_t, we obtain a depth map prediction $D(X_t)$ and aleatoric variance prediction $\sigma_a^2(X_t)$ for this image. In order to merge measurements that are multiple views of the same points in 3D space, we maintain a point cloud of C 3D points and associated uncertainty from previous predictions. To identify pixels that are a new view of a point we have seen before, we project back the point cloud on to the image plane and add points we are seeing for the first time to the point cloud. To combine the measurements of multiple views of the same

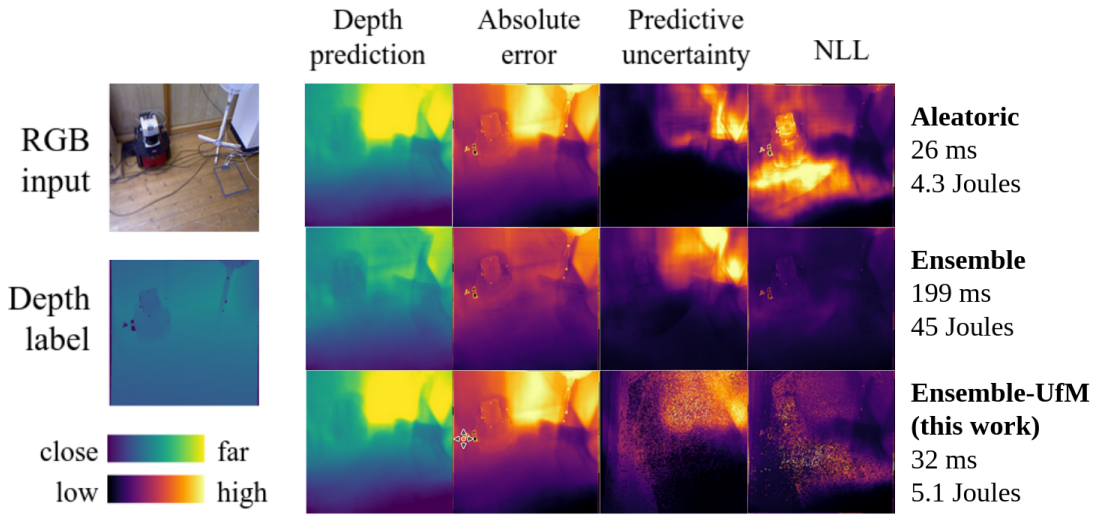


Figure 4: Uncertainty estimation comparison for an aleatoric network, ensemble, and UfM applied to ensembles on an out-of-distribution cropped example from the TUM RGBD [54] dataset. Lower NLL indicates better uncertainty quality. ensemble-UfM is able to achieve close to similar NLL as ensemble at a fraction of the cost.

point in 3D space c, like in ensembles [34], we treat the depth and aleatoric variance predicted per pixel as a Gaussian and combine DNN predictions as a new Gaussian with the mean and variance of a mixture of Gaussians incrementally each time we see an additional view of the same point in 3D space. The mean aleatoric variance is taken to be the aleatoric variance $\sigma^2_{a,c}$ and the variance of the depth predictions is the epistemic variance $\sigma^2_{e,c}$ for point c in 3D space, which can be projected back to the current image plane to find $\sigma^2_a(X_t)$ and $\sigma^2_e(X_t)$. Note, the correspondences in UfM are not guaranteed to be correct since we rely on the previous predicted DNN depth; empirically, we find the noisy correspondences are sufficient for obtaining ensemble-like uncertainty quality and allow us to keep the overhead lightweight. The formal problem definition and details of the algorithm can found in Sudhakar et al [55].

Experimental evaluation: Even though there is no ground-truth for uncertainty, a "well-calibrated" uncertainty estimate would be high when error is high and low when error is low. Negative log-likelihood (NLL) captures this trend, as given by

$$\begin{aligned} NLL = & \frac{1}{2(\sigma^2_e(X_t) + \sigma^2_a(X_t))}(D_{gt}(X_t) - D(X_t))^2 \\ & + \frac{1}{2} ln(2\pi(\sigma^2_e(X_t) + \sigma^2_a(X_t))), \end{aligned} \tag{1}$$

where $D_{gt}(X_t)$ is the groundtruth depth for current image X_t; a lower NLL can indicate better uncertainty quality. Figure 1 shows the pixel-wise NLL for the baselines 1) aleatoric only, 2) ensemble, and 3) ensemble-UfM. As we can see, ensemble-UfM obtains similar NLL to ensemble at a fraction of the latency and energy since it only needs to run

one DNN per image. As we can see, ensemble-UfM obtains close to ensemble uncertainty quality, at a fraction of the cost.

3.3 DecTrain: Deciding When to Train a Monocular Depth DNN

While using a monocular depth DNN can be more energy efficient and has a smaller form factor than traditional bulky and high-power physical depth sensors such as LiDAR or active IR stereo [26, 56], DNNs are prone to accuracy degradation on images that differ from those of the training distribution in a range of domains [29, 3, 5, 31]. One solution is online training, where the model continuously learns and adapts during deployment using self-supervised techniques. While this can significantly improve accuracy, it is computationally expensive, especially on resource-constrained platforms like mobile robots and drones. Performing online training at every timestep is often unnecessary, as some frames contribute little to improving accuracy. This creates a trade-off between computational efficiency and model adaptation. In this work, we propose a new method called DecTrain that decides whether to train a monocular depth DNN at each timestep based on when the potential accuracy improvement is worth the computational cost of training. By balancing the accuracy and compute cost, DecTrain enables low-cost online training for a smaller DNN to have competitive accuracy with a larger, more generalizable DNN at a lower overall computational cost.

Methodology: DecTrain selectively trains based on a balance between accuracy improvement and computational cost. The problem is framed as a Markov Decision Process (MDP), where the state consists of factors like self-supervised loss, model uncertainty, and environmental characteristics. At each timestep, the model can choose between two actions: training or not training, with a reward function that considers the trade-off between computational cost and potential accuracy gain. To make this decision, DecTrain predicts the utility $\widetilde{\mathcal{U}}_t$ of training at the current timestep t by assessing two key factors: the margin to improve and the ability to improve, as shown in Fig. 6. The former refers to the gap in accuracy between how well the monocular depth DNN is currently performing and how well it could perform with perfect guidance, and the latter refers to the ability of the self-supervision to guide the DNN weights in the correct direction to improve accuracy. The margin to improve is estimated using the self-supervised loss and epistemic uncertainty, which captures the uncertainty in the model weights that can be reduced with training. Meanwhile, the ability to improve is inferred from aleatoric uncertainty, which reflects noise in the data that training cannot resolve, as well as scene texture and motion cues that indicate whether the online training would be effective. DecTrain greedily decides whether to train at each timestep t based on the reward $R_t = \frac{-1}{\alpha} + \widetilde{\mathcal{U}}_t$ at t, where $\frac{-1}{\alpha}$ represents the cost of training with a user-defined α to balance training cost with potential accuracy improvement. If the potential accuracy improvement is worth the compute cost, *i.e.* $R_t > 0$, DecTrain would decide to train at the current timestep. The decision-making process is powered by a lightweight decision DNN, which is initially trained offline on pre-collected datasets and then updated online during deployment. This allows DecTrain to continuously refine its training decisions in response to new environments. The formal problem definition and

details of the algorithm can found in RA-L 2025 [16].

Experimental evaluaton: We evaluate DecTrain on 133 sequences from out-of-distribution indoor and outdoor datasets (ScanNet [8], SUN3D [58], KITTI-360 [36]). Compared to performing online training at all timesteps, DecTrain is able to reduce the percentage of timesteps we train to 30-58% while maintaining accuracy on 10 representative experiments. We also show that low inference cost DNNs using DecTrain can achieve competitive accuracy (4-6% higher accuracy) and lower computational cost (17-57% lower GFLOPs) compared to a high inference cost state-of-the-art DNN across 100 out-of-distribution sequences. Showing competitive accuracy and computational savings compared to performing online training at all timesteps, DecTrain highlights the importance of deciding wisely and efficiently to deliver a higher accuracy at lower cost when adapting to a new environment.

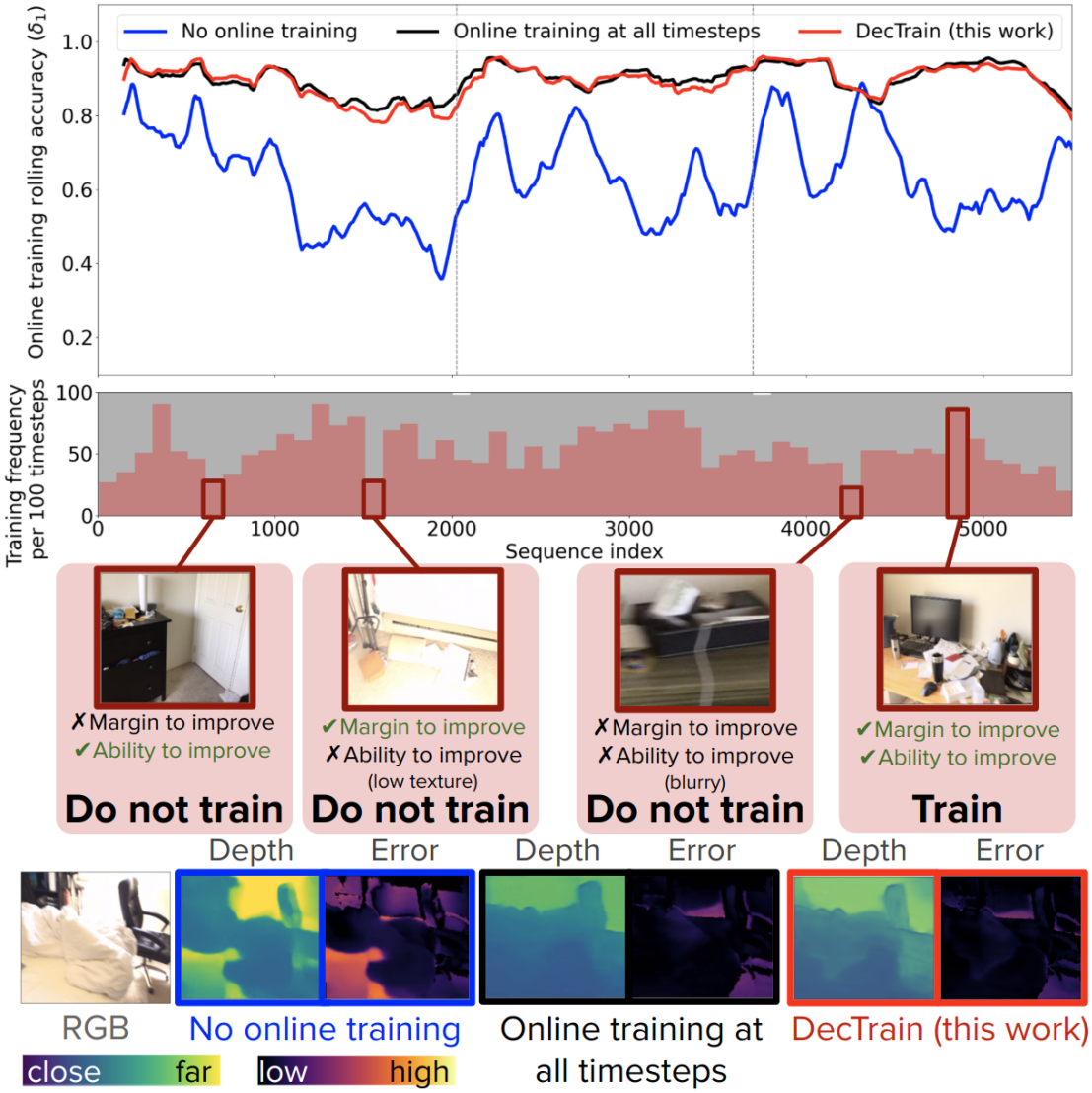


Figure 5: DecTrain (red) decides when to perform online training based on *margin to improve* (visualized by the gap between the blue and black lines) and *ability to improve* (visualized by the texture and sharpness in the image). Compared to the baseline of online training at all timesteps (black) or no timesteps (blue), DecTrain maintains the accuracy improvement of adaptation while training on only a subset of the timesteps (dashed lines denote new sequence).

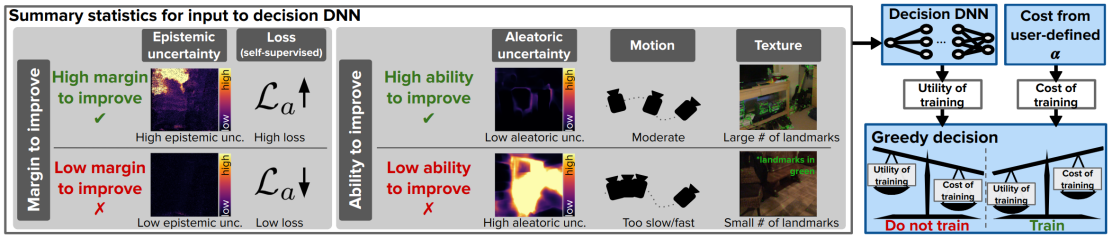


Figure 6: DecTrain overview: at each timestep, the decision DNN takes inputs relevant to the margin and ability to improve to predict the utility of training, which is compared to the cost of training to decide when to train the monocular depth DNN.

4 Conclusion

In this report, we presented results that will help reduce the compute cost of autonomous driving systems. We first presented a new Gaussian splatting based SLAM system that is able to reduce the memory overhead of the system by 94x. We then presented a new way to train a monocular depth DNN on the fly, which will allow us to reduce the compute cost of the system while maintaining accuracy.

These two contributions show that the compute cost of autonomous driving systems, while still significant, can be reduced by at least two orders of magnitude using custom chips designed for autonomous driving. In fact, this is the trend that we observe in the industry, as virtually all OEMs are designing custom chips for advanced driver assistance systems (ADAS) and autonomous driving. We expect that this trend will continue in the future, as the compute cost of these systems continues to decrease.

References

[1] Ashutosh Agarwal and Chetan Arora. Attention attention everywhere: Monocular depth prediction with skip attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5861–5870, 2023.

[2] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *arXiv preprint arXiv:1910.02600*, 2019.

[3] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in Neural Information Processing Systems*, 33:14927–14937, 2020.

[4] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.

[5] Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision workshops*, pages 0–0, 2019.

[6] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

[7] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[9] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[10] Aditya Dhawale and Nathan Michael. Efficient parametric multi-fidelity surface mapping. In *Robotics: Science and Systems (RSS)*, volume 2, page 5, 2020.

[11] Kevin Doherty, Tixiao Shan, Jinkun Wang, and Brendan Englot. Learning-aided 3-d occupancy mapping with bayesian generalized kernel inference. *IEEE Transactions on Robotics*, pages 1–14, 2019. doi: 10.1109/tro.2019.2912487. URL `https://doi.org/10.1109/tro.2019.2912487`.

[12] Benjamin Eckart, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz. Accelerated generative models for 3d point cloud data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5497–5505, 2016.

[13] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[14] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[15] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[16] Zih-Sing Fu, Soumya Sudhakar, Sertac Karaman, and Vivienne Sze. Dectrain: Deciding when to train a monocular depth dnn online. *IEEE Robotics and Automation Letters*, 2025.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[18] Yuan Gao and Wei Dong. An integrated hierarchical approach for real-time mapping with gaussian mixture model. *IEEE Robotics and Automation Letters*, 2023.

[19] Kshitij Goel, Nathan Michael, and Wennie Tabib. Probabilistic point cloud modeling via self-organizing gaussian mixture models. *IEEE Robotics and Automation Letters*, 8(5):2526–2533, 2023.

[20] Vitor Guizilini and Fabio Ramos. Towards real-time 3d continuous occupancy mapping using hilbert maps. *The International Journal of Robotics Research*, 37(6): 566–584, 2018.

[21] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[22] Jonathan Heek and Nal Kalchbrenner. Bayesian inference for large scale image classification. *arXiv preprint arXiv:1908.03491*, 2019.

[23] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[24] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

[25] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular stereo and rgb-d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21584–21593, 2024.

[26] *D400 Series Product Family Datasheet*. Intel, 7 2023.

[27] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

[28] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[29] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977*, 2017.

[30] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[31] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[32] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[34] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[35] Peter Zhi Xuan Li, Sertac Karaman, and Vivienne Sze. Memory-efficient gaussian fitting for depth images in real time. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8003–8009. IEEE, 2022.

[36] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022.

[37] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5162–5170, 2015.

[38] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

[39] Hidenobu Matsuki, Edgar Sucar, Tristan Laidow, Kentaro Wada, Raluca Scona, and Andrew J Davison. imode: Real-time incremental monocular dense mapping using neural field. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4171–4177. IEEE, 2023.

[40] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[41] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18039–18048, 2024.

[42] Raul Mur-Artal and Juan Tardos. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015. doi: 10.15607/RSS.2015.XI.041.

[43] Jia Ning, Chen Li, Zheng Zhang, Chunyu Wang, Zigang Geng, Qi Dai, Kun He, and Han Hu. All in tokens: Unifying output space of visual tasks via soft token. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19900–19910, 2023.

[44] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[45] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[46] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

[47] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[48] Cormac O'Meadhra, Wennie Tabib, and Nathan Michael. Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robotics and Automation Letters*, 4(2):2015–2022, 2018.

[49] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[50] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021.

[51] Jari P Saarinen, Henrik Andreasson, Todor Stoyanov, and Achim J Lilienthal. 3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments. *The International Journal of Robotics Research*, 32(14):1627–1644, 2013.

[52] Ashutosh Saxena, Sung Chung, and Andrew Ng. Learning depth from single monocular images. *Advances in neural information processing systems*, 18, 2005.

[53] Shobhit Srivastava and Nathan Michael. Efficient, multifidelity perceptual representations via hierarchical gaussian mixture models. *IEEE Transactions on Robotics*, 35(1):248–260, 2018.

[54] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[55] Soumya Sudhakar, Vivienne Sze, and Sertac Karaman. Uncertainty from motion for dnn monocular depth estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8673–8679. IEEE, 2022.

[56] *Velodyne Lidar Puck*. Velodyne, 2019.

[57] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019.

[58] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE international conference on computer vision*, pages 1625–1632, 2013.

[59] Xuan Yang, Liangzhe Yuan, Kimberly Wilber, Astuti Sharma, Xiuye Gu, Siyuan Qiao, Stephanie Debats, Huisheng Wang, Hartwig Adam, Mikhail Sirotenko, et al. Polymax: General dense prediction with mask transformer. *arXiv preprint arXiv:2311.05770*, 2023.

[60] Youmin Zhang, Fabio Tosi, Stefano Mattoccia, and Matteo Poggi. Go-slam: Global optimization for consistent 3d instant reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3727–3737, 2023.

[61] Weiming Zhi, Lionel Ott, Ransalu Senanayake, and Fabio Ramos. Continuous occupancy map fusion with fast bayesian hilbert maps. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4111–4117. IEEE, 2019.

[62] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. In *2024 International Conference on 3D Vision (3DV)*, pages 42–52. IEEE, 2024.