Memory-Efficient Real Time Many-Class 3D Metric-Semantic Mapping

Vallabh Nadgir* Joao Marcos Correia Marques* Kris Hauser

***Abstract*— Metric-semantic 3D mapping is the process of creating class-labeled 3D maps by fusing the information from images captured by a moving camera. The memory usage required by standard solutions grows linearly with the number of semantic classes being considered, which can pose a bottleneck in large and many-class scenes. This paper proposes two novel methods for compressing the memory used by semantic fusion: calibrated top-k histogram and encoded fusion. The first method maintains, for each voxel, only the counts of the k most likely classes, while the second method uses a neural network to encode all-class probability vectors into a k-dimensional latent space in which per-voxel fusion is performed. The fused result is then decoded, at query time, using another neural network. Experiments show that both methods preserve map accuracy and calibration even at low values of k, and per-voxel memory usage is linear in k. The proposed methods can achieve real-time semantic fusion with 150 classes on commodity GPUs in building-scale scenes where prior approaches run out of memory.**

I. Introduction

Semantic segmentation is a common task in robotic perception that involves labeling each part of a scene as one of several object classes. While image segmentation has been a longstanding problem in computer vision, many robotic applications require 3D semantic segmentation in which a segmented 3D map is built by accumulating the semantic information from multiple views [1, 2]. Although there are some offline methods that segment a 3D map after a dataset has been gathered [3], robotic applications like SLAM require incremental methods that build the scene geometry and its segmentation online as more camera views are obtained [2, 4, 5]. The common *semantic fusion* approach first segments each input image and then employs recursive estimation to update the 3D map and its segmentation incrementally and in real time [1, 6, 7].

The time and memory complexity of standard fusion techniques scales with the number of classes due to the need to store categorical distributions across voxel classes, and when the map or the number of classes is large, this can overwhelm GPU memory. GPU memory is a particularly limiting factor for mobile robots, drones, and other mobile devices that operate over long distances under battery power. Technological trends are also scaling segmentation models and benchmarks from a few dozen common classes [8, 9] to hundreds of long-tail classes [10]. Moreover, memory usage will also be an important bottleneck in open-world semantic segmentation, since open-world feature encodings are often in the hundreds of dimensions. Our work investigates compressed representations of semantic maps that enable fusion with bounded per-voxel memory usage as the number of classes grows large.

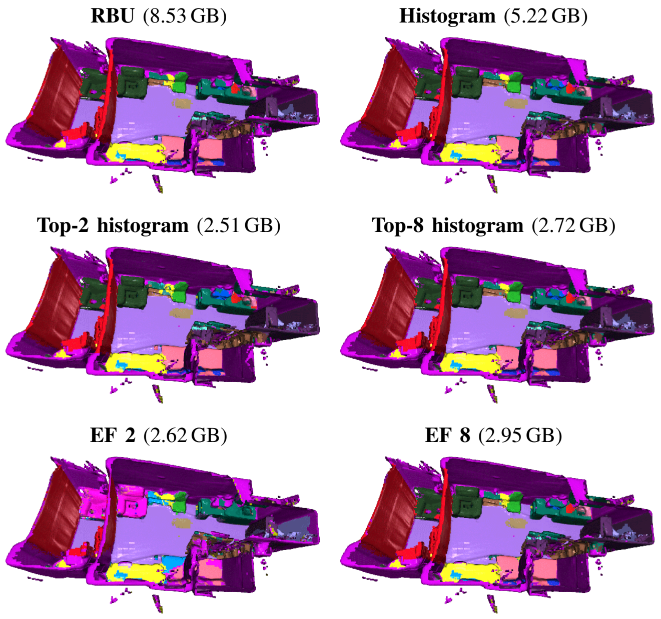


Fig. 1: Maps and GPU memory usage of various semantic fusion methods on a small scene from ScanNet++ considering 150 classes. Recursive Bayesian Update and Histogram consume much more VRAM than our approaches, Calibrated Top-k histogram and Encoded Fusion (EF) while obtaining similar reconstruction quality. Note that the Segformer segmentation model consumes 1.3 GB and the geometric map consumes an additional 1.15 GB.

The challenge of compression is to preserve map accuracy and confidence calibration while bounding memory usage. A voxel can be classified with different labels as it is observed from different viewpoints, and discarding information may lead to accuracy degradation or overconfident uncertainty estimates. Note that *confidence* refers to an approximation of the probability that the predicted label is correct [11, 12, 13], and a more accurate confidence value is said to be well calibrated. In practice, we find that a single voxel is usually labeled as a small number of distinct classes over all the viewpoints that observe it. For example, on the 21-class ScanNet [8] dataset, a finetuned Segformer [14] model predicted 4 or fewer classes per voxel over more than 99% of voxels. We make use of this observation in the two compression methods presented in this paper.

The first method is the *calibrated top-k histogram*, which collapses evidence for classes not represented in the top k represented classes into a single scalar. We prove that this method approximates a full histogram with bounded error, in

* denotes equal contribution.
School of Computing and Data Science, University of Illinois at Urbana-Champaign, IL, USA. {vnadgir2, jmc12, kkhauser}@illinois.edu. This project was partially supported by USDA/NIFA Grant #2020-67021-32799 and NSF Grant #2409661 .

which the bound is zero in common cases, and in practice the error is low even for small values of k. The second method is *encoded fusion*, in which a voxel's categorical distribution is compressed to a fixed-dimensional feature vector using a neural network. The encoder and a corresponding decoder are trained end-to-end through the fusion process to yield high accuracy and well-calibrated predictions. Compared to top-k histogram, encoded fusion can achieve better confidence calibration on some datasets but requires additional training. Both methods require only minor adjustments to existing semantic fusion pipelines.

Experiments on the ScanNet and ScanNet++ datasets demonstrate that both methods add little running time compared to baselines while reducing memory usage (Figure 1). Most importantly, even at small k, our methods retain high accuracy and calibration of the fused maps. Applied to the BS3D building-scale dataset with up to 150 classes [15], our methods enable real-time 3D segmentation with memory savings of over 70%. An implementation of our method is available at https://github.com/uiuc-iml/memory-efficient-3d-semantic-mapping

II. Related Work

A variety of semantic 3D mapping methods have been developed over the years. We focus our attention on incremental techniques that update the geometric and semantic map after each frame [2, 4, 5]. In contrast, non-incremental techniques calculate the map after the frame sequence has been produced and can use representations like Neural Radiance Fields [3] and Gaussian Splatting [16] to perform better geometric regularization and segment-level reasoning. But for applications in robotics and VR, real-time methods that scale to large spaces are preferred because they are useful for online planning and state estimation. Our work also uses the popular voxel-based truncated signed distance field (TSDF) representation [2, 6, 7, 17], although our work could also be applied to point or surfel clouds [1, 4, 18, 19] without significant modification. We also focus on compressing per-voxel memory demands rather than increasing accuracy, since voxel-level classification is a first step toward segment-level regularization [2, 5] and hierarchical, object-based 3D mapping [18, 19, 20, 21, 22]. Our technical contributions could easily be incorporated into more sophisticated semantic mapping systems.

Our proposed calibrated top-k histogram method is inspired by prior work in instance (panoptic) segmentation. To handle the unbounded instance IDs during fusion, Panoptic Fusion [2] and maintains, per-voxel most-likely instance ID and a weight associated with this ID. This weight is decremented when the predicted ID is not observed in a new scan, and is replaced when the weight drops below 0. This is similar to a naive version of our top-k strategy in the case that $k = 1$, but fails to provide confidence estimates [13]. For assigning semantics to the instance, it still retains a categorical distribution over all classes. PanopticNDT [23] extends this idea by keeping top 16 instance ids, and drops the least likely instance on a "miss." The paper generalizes to arbitrary $k > 1$, proves that weight decrementing yields bounded histogram error, and introduces a confidence correction step that improves calibration.

Our encoded fusion technique is related to learning-based aggregation [3, 24, 25], in which the fused voxel representation is related to image segmentation by a learned map. However, NeRF-based prior work [3, 24] is not incremental, and the model is optimized to segment the volume covered by a given image sequence. Similarly, recent Gaussian Splatting work [26] proposes to learn a per-scene encoder-decoder which performs dimensionality reduction on open-world language features, storing only the reduced-dimension hidden layer and decoding it at the end. This approach, however, is not suitable for online real-time applications, as it requires training scene-specific encoders. In contrast, our work learns a compressed representation that is suited for fast incremental updates, and is segmentation-model specific rather than scene-specific . DA-RNN [25] is also incremental and trains an RNN to fuse the results from subsequent scans. However, it is trained on 3-frame batches which can easily lose historical dependencies, whereas our approach uses mean-aggregation and can accommodate hundreds of frames. Moreover, their latent vector is 64-D when only 7 classes are represented, leading the map to be enlarged rather than compressed.

III. Method

We consider the semantic fusion problem, which takes as input a series of RGB-D color/depth images (c_i, d_i), and camera poses T_i relative to the world frame, $i = 1, \ldots, n$, to produce a 3D map $\mathcal{V} = (v_1, \ldots, v_m)$ consisting of voxels $v_j \in \mathcal{V}$, each of which contains a TSDF value, a count N_j of observations of the voxel (often called weight) and a categorical distribution P_{v_j} over C semantic classes. Semantic fusion methods propose to efficiently update the map from the $(n-1)$-th step to the n-th step using only the information from the n-th image.

Our work addresses the problem that standard semantic fusion methods scale linearly with C in both time and space due to the need to store and update per-voxel categorical distributions. Specifically, P_{v_j} is usually represented by a likelihood vector $l_j \in \mathbb{R}^C$ satisfying $l_j \geq 0$ and $\sum l_j^c = 1$. In contrast, our work introduces compressed representations of P_{v_j} that use a constant size independent of C.

A. Background on 3D semantic fusion

Semantic fusion uses a semantic segmentation network $\sigma_s(c_i, d_i)$ that takes color and depth $(c_i,\ d_i)$ as input and produces a semantic segmentation $s_i \in \mathbb{R}^{W \times H \times C}$, which represents a categorical distribution over the C considered classes for each pixel. The fusion step uses c_i, d_i, and s_i to update $\mathcal{V}$ by projecting the ray for each pixel into 3D scene space and updating the metric map and semantic likelihoods l_j for the corresponding voxel v_j. We follow the approach of Niessner et al. [27] with the spatial hashing strategy of Dong et al. [28] for metric mapping.

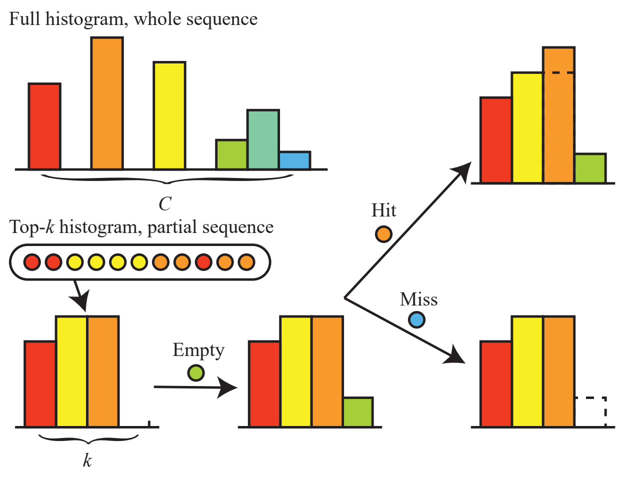

Full histogram, whole sequence
C
Top-k histogram, partial sequence
k
Empty
Hit
Miss

Fig. 2: Illustrating the top-k histogram technique, $k = 4$. Three outcomes are possible when each new label is observed: empty, hit, or miss. In the empty case, fewer than k distinct labels have been observed and the new label is added. For a hit, the count of the existing label is incremented. For a miss, the count of the least likely existing label is decremented.

Semantic fusion methods differ by their approach to updating class likelihoods l_j for an observed voxel v_j. For notational convenience, let us drop the subscript j, and let us identify each image in which the voxel of interest was observed. In each such image, let $\{x_1^c, \ldots, x_N^c\}$ denote the semantic likelihood of class $c = 1, \ldots, C$ at the pixel corresponding to the voxel of interest, and denote $\{x_1^\star, \ldots, x_N^\star\}$ as the most likely image class labels of the voxel.

The original and most common approach is to perform a Recursive Bayesian Update (RBU) [1, 2, 6, 7, 12, 17, 29], which assumes that each observation is conditionally independent given the class, with $l^c = \frac{1}{\eta}\prod_{i=1}^N x_i^c$ where η is a normalization constant to ensure class likelihoods sum to 1. In practice, this is implemented as a recursive sum of logs.

Another common approach is Naïve Averaging (NA) [5], which approximates l_n^c as:

$$l^c = \frac{1}{N}\sum_{i=1}^{N} x_i^c \text{ or, recursively, } l^c \leftarrow l^c + \frac{x_N^c - l^c}{N} \tag{1}$$

Finally, the histogram method [20, 21], stores a histogram $h^c \in \mathbb{N}$ of the number of times the class c was the most likely prediction in the image, and performs the update

$$h^c \leftarrow h^c + \mathbb{I}[c = x_N^\star] \tag{2}$$

where $\mathbb{I}$ is an indicator function. Both histogram and NA were shown to be better calibrated than RBU [13]

Note that RBU, naïve averaging, and histogram updates use $O(C)$ memory per voxel to store semantics. RBU and NA require storing floating-point numbers, whereas a histogram can store integers and aggressively truncate to save memory, e.g., an 8 or 16 bit integer per class.

B. Method 1: Calibrated Top-k Histogram (CTKH)

Rather than storing the entire set of histogram counts, CTKH maintains counts of the k most represented classes. The rationale behind this approach is that semantic segmentation models are often consistent on the most likely prediction, and tend to predict a voxel as belonging only to a handful of classes across an entire sequence.

For each voxel, CTKH stores two vectors: a *class* vector and a *counts* vector. The class vector $\bar{c} \in \{1, \ldots, C\}^k$ includes k class indexes and the counts vector $\bar{h}$ stores k integers, one for each class in $\bar{c}$.

When a label $c = x_N^\star$ is encountered, we first examine if the label exists in $\bar{c}$. If $c \in \bar{c}$ (a *hit*), then we increment the associated count entry in $\bar{h}$. Otherwise, if any entry in the counts vector is zero (the *empty* condition), then the corresponding entry in the class vector $\bar{c}$ is set to c and its count is incremented. If no entry is zero (a *miss*), we decrement the class count of the lowest value of $\bar{h}$. If this value drops to 0, then the corresponding entry of $\bar{c}$ becomes empty and can become filled in the next observation. Figure 2 illustrates this process.

To estimate P_v, we use a *calibration corrected* version of the frequency estimates which produces a better estimate of class probabilities accounting for misses. Define the pre-correction frequency estimate $\tilde{l}$ as 0 for all classes not in $\bar{c}$, and $\tilde{l}^{\bar{c}_i} = \bar{h}_i/(\sum_l \bar{h}_l)$ for $i = 1, \ldots, k$. The probability mass not accounted for in the truncated histogram is $\alpha = 1 - \frac{\sum_l \bar{h}_l}{N}$. The calibration correction compensates for this mass by adjusting $\tilde{l}$ toward the uniform distribution via:

$$P_v^c = (1-\alpha)\tilde{l}^c + \alpha/C. \tag{3}$$

Note that we do not explicitly store this as a likelihood vector, but rather evaluate it at query time. Experiments in Figure 3 demonstrate that CTKH is better calibrated than a naïve truncated histogram (i.e., $\alpha = 0$) at lower values of k, dropping as k grows and the number of "misses" drops.

We provide three guarantees about CTKH that hold for any k or observation order. In these proofs, we denote $\tilde{h} \in \mathbb{N}^C$ as the expanded class histograms produced by CTKH and h as the true histogram that would be produced by the histogram method. Moreover, we denote the discrepancy as $\beta = h - \tilde{h}$. Let l_{CTKH} be the likelihood vector from (3) and $l_H = h/N$ be the likelihood vector from a true histogram.

Theorem 1: If k or fewer classes are observed for a voxel, then $\tilde{h} = h$ and $l_{CTKH} = l_H$. The proof is trivial.

Lemma 1: β increments two distinct entries for each "miss": the class that is not being added to the top-k and the least observed class in the top-k. Hence, $\beta^c \le \sum_{k \ne c} \beta^k, \forall c$.

Theorem 2: If a voxel is labeled by a class $c^\star = x_i^\star$ in a majority of images (i.e., $h^{c^\star} > N/2$), then CTKH labels it correctly and its count matches that of the histogram (i.e., $\arg\max_c \tilde{h}^c = h^{c^\star}$).

Proof: Since $h^{c^\star} > N/2$, we have that $h^{c^\star} > \sum_{c \ne c^\star} h^c$. Applying the inequality to $\tilde{h}^{c^\star} = h^{c^\star} - \beta^{c^\star}$, we get that:

$$\tilde{h}^{c^\star} > \textstyle\sum_{c \ne c^\star} h^c - \beta^{c^\star} \overset{\text{Lemma 1}}{\ge} \sum_{c \ne c^\star} h^c - \sum_{c \ne c^\star} \beta^c = \sum_{c \ne c^\star} \tilde{h}^c. \tag{4}$$

Since $\tilde{h}$ is nonnegative, $\tilde{h}^{c^\star} \ge \tilde{h}^c$ for all c as desired. ■

Theorem 3: The difference between the likelihood determined by CTKH and the histogram is bounded by $\|l_{CTKH} -$

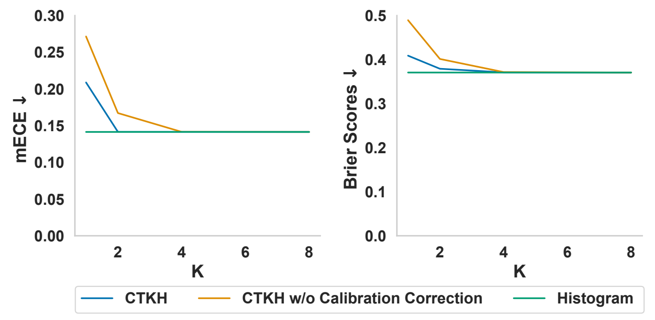


Fig. 3: mECE [13] and Brier scores (lower is better) for CTKH with and without calibration correction on a 10-scene sample of ScanNet. Lower values are better. Calibration correction leads to improved confidence estimates at low values of k.

$l_H\| = \frac{1}{N}\|\beta\| \leq \frac{D\sqrt{2}}{N}$ where $D = \frac{1}{2}\mathbb{1}^T\beta$ is the number of "misses".

Proof: The sum of top-k histogram entries is $\sum_l \tilde{h}_l = N - \mathbb{1}^T\beta = N - 2D$, and $\alpha = 2D/N$. Replacing this and expanding, we get

$$l_{CTKH} - l_H = \frac{\tilde{h}}{(N-2D)}(1-\alpha) + \frac{\alpha\mathbb{1}}{C} - \frac{h}{N}. \quad (5)$$

Using the definition of α and performing a bit of algebra,

$$\begin{aligned} l_{CTKH} - l_H &= \frac{(h-\beta)}{N} + \mathbb{1}\frac{2D}{(NC)} - \frac{h}{N} \\ &= -\frac{\beta}{N} + \mathbb{1}\frac{2D}{NC}. \end{aligned} \quad (6)$$

Taking the squared norm, we have

$$\begin{aligned} \|l_{CTKH} - l_H\|^2 &= \frac{1}{N^2}(\mathbb{1}\frac{D}{C} - \beta)^T(\mathbb{1}\frac{D}{C} - \beta) \\ &= \frac{1}{N^2}(4\frac{D^2}{C} - 2\beta^T\mathbb{1}\frac{D}{C} + \|\beta\|^2) \end{aligned} \quad (7)$$

because $\|\mathbb{1}\|^2 = C$. Since $\beta^T\mathbb{1} = 2D$, the first two terms in the sum cancel out and hence $\|l_{CTKH} - l_H\|^2 = \frac{1}{N^2}\|\beta\|^2$. Now, note that no term in β can exceed D and its entries must satisfy $\mathbb{1}^T\beta = 2D$. With these constraints, we can bound the norm of β to $D\sqrt{2}$. ■

The worst-case discrepancy is observed in the following case. Assume that among all observations of a voxel, class k and class $k+1$ are the most likely classes, observed with nearly 50% likelihood and that classes 1 to $(k-1)$ are observed twice. Then, suppose the first $2(k-1)$ observations are of classes 1,...,$k-1$, then we observe a long sequence alternating between classes k and $k+1$. The intermediate sequence will see the most likely classes "fighting" over the k'th spot in the histogram, repeatedly kicking each other out. The resulting distribution will be uniform over classes 1,...,$(k-1)$ rather than having peaks at k and $k+1$. Calibration correction helps reduce the discrepancy, but the output will be nearly uniform. We suspect that such pathological cases are rare, which is reflected in our experimental results.

C. Method 2: Encoded fusion

The encoded fusion method uses an encoder-decoder neural network structure to encode the class probability vector into a low-dimensional latent space, as shown in Figure 4. We then compute the running mean of the latent vectors using the NA fusion technique and use the decoder on the averaged feature vectors to extract the posterior over all classes.

Specifically, we store encoder and decoder networks $f^{enc}(\cdot;w^{enc}) : \mathbb{R}^C \rightarrow \mathbb{R}^k$ and $f^{dec}(\cdot;w^{dec}) : \mathbb{R}^k \rightarrow \mathbb{R}^C$ along with their weights w^{enc} and w^{dec}. The fused result is a per-voxel vector $\tilde{q}_i \in \mathbb{R}^k$ that is given by the average of latent vectors at time i:

$$\tilde{q}_n = \frac{1}{n}\sum_{i=1}^{n} f^{enc}(x_i; w^{enc}). \quad (8)$$

The average is maintained by online averaging, and the resulting class likelihoods are $l = f^{dec}(q_n; w^{dec})$.

While fusion is conceptually straightforward using this technique, training the encoder-decoder networks requires a bit more care. Although a standard autoencoder applied to probability vectors might be a reasonable first attempt, averages of latent vectors are not guaranteed to meaningfully approximate averaged probabilities. So, we train the entire encoder-fusion-decoder pipeline end-to-end with image probabilities as input and voxel probabilities as output.

The procedure is similar to the semantic fusion training method proposed in [13]. A training set is given by a set of image segmentations $\{s_i \,|\, i = 1, ..., N\}$, their mapping to voxels, as well as the target voxel posteriors $\{\tilde{l}_j^{out} \in \mathbb{R}^C\}$. If ground truth voxel labels are available, using the one-hot encoded vectors as targets could enable the encoder-decoder to learn an improved fusion strategy to boost accuracy beyond standard fusion. However, ground-truth labels may not be available, and moreover this runs the risks of overfitting and overconfidence since labeled 3D data are scarce. Instead, we opt to use the results of standard NA fusion as psuedolabels. This strategy inherits the favorable calibration properties of NA, and it requires no ground truth labels, which allows our method to train on a larger data set.

Therefore, for each voxel we preprocess the set of image-associated class probabilities $x_{ij}, i = 1, \ldots, N_j$ where N_j is the number of image pixels associated with voxel j. We set $\tilde{l}_j^{out} = \frac{1}{N_j}\sum_{i=1}^{N_j} x_{ij}$, and train to minimize the loss:

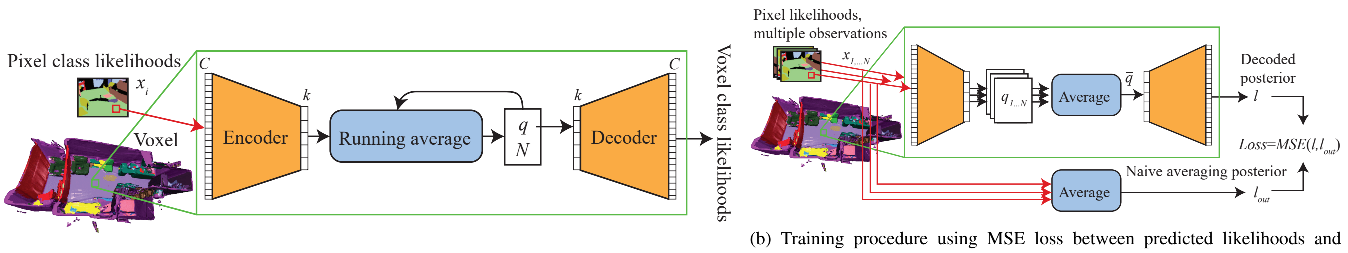
$$\ell\left(\tilde{l}_j^{out}, f^{dec}\left(\left(\frac{1}{N_j}\sum_{i=1}^{N_j} f^{enc}(x_{ij}; w^{enc})\right), w^{dec}\right)\right) \quad (9)$$

over all voxels. We use mean squared error for the loss, as illustrated in Figure 4.

We remark that training is performed by preprocessing a set of scenes whose image and voxel class distributions are assumed to be representative of scenes encountered in testing. This is because learning will tend to discard unlikely combinations of classes in its feature encoding.

D. Time Complexity and Memory Footprint Analysis

CTKH stores the class vector and counts vector as 16 bit integers, leading to $4k$ bytes stored per voxel for semantics, as opposed to the standard histogram's $2C$ bytes. Operations to update the histogram are performed in parallel across all voxels updated in a single timestep. These operations (and their time complexities) are: checking if an observed class is



(a) Encoded Fusion architecture during reconstruction.

(b) Training procedure using MSE loss between predicted likelihoods and pseudo-labels from Naïve Averaging.

Fig. 4: (a) Architecture and (b) training procedure of the Encoded Fusion model. The decoder block is only used at query time and not during reconstruction.

in the class vector (O(k)) and either incrementing its count vector (O(1)) or finding the lowest entry in the counts vector (O(k)), decrementing it (O(1)) and removing its entry in the class vector if the count reaches zero (O(1)). Since this is done for all voxels, the final time complexity of this update is $O(kN_v)$, where N_v is the total number of voxels being updated. To extract the most-likely class of a voxel and its confidence is O(k), whereas computing the all-class posterior is O(C).

Our encoded fusion implementation stores 32-bit floating point vectors, also leading to $4k$ bytes per voxel. Running times are variable due to the choice of the encoder network. The decoder is only run during probability vector extraction. The query of a most-likely class and its confidence does require $O(C)$ space per voxel, but this storage is transient and can be allocated and deallocated for each voxel block.

IV. Experiments

A. Implementation and Evaluation Hardware

We implement both EF and CTKH with PyTorch and the sparse voxel block grids TSDF fusion [27] pipeline of Open3D version 0.18.0 [30]. CTKH performs operations in parallel on the GPU with all branching implemented via masks. Unless otherwise specified, we use a voxel resolution of 2.5 cm, 8×8×8 blocks, and initialize the block hash map capacity with 17,500 blocks. As more scans are acquired and the existing block capacity is exceeded, the hash map size is doubled until VRAM is exhausted. Timing and memory experiments are performed on an AMD Ryzen 9 5950x 16-core processor with Nvidia GeForce RTX 4090 and 24 GB VRAM running Ubuntu 22.04, while some accuracy and calibration experiments were performed using A100 GPUs in the NCSA Delta cluster [31].

B. Datasets, Segmentation, and Encoder Details

We validate our proposed methods in 3 different datasets: ScanNet [8]. ScanNet++ [10] and BS3D [15]. ScanNet has a 21-class annotation (20 classes + background), ScanNet++ has a 101-class (100 + background) benchmark and BS3D has no ground-truth semantic annotations, but has very large scale environmental scans.

Our experiments consider two segmentation models. One transformer-based architecture, Segformer [14] segformer-b4-finetuned-ade-512-512 from Hugging Face, which is trained on the ADE20K dataset [32], and a CNN-based model, ESANET [33]. In ScanNet experiments, we fine-tune Segformer and ESANET to have a 21-class output on 79 scenes from the training set, with 21 scenes used for validation. For experiments on ScanNet++, we split the original training and validation scenes into training (746), validation (80), and evaluation sets (80). The original hidden test set is not used in our experiments, since it lacks ground-truth annotations required for calculating calibration metrics. We finetune Segformer to have 101 outputs using the training scenes (and the validation scenes to early stop finetuning) and do not use ESANET, as it had not been shown to perform long-tail segmentation well.

The EF-k models use a 4-layer MLP for each of the encoder and decoder with hidden sizes 256, 128, 64 and k (with inverted order for the decoder). These models are trained on voxels and semantic segmentations of the scenes used for validation of the respective semantic segmentation finetuning datasets. We split these voxels and semantic images into train/val/test splits with a 80:10:10 ratio. These models were trained with the ADAM optimizer and a 10^{-3} learning rate.

C. Computational Performance

To evaluate computational efficiency, we reconstruct 70 scenes from ScanNet++ using RBU, NA, Histogram, CTKH and EF, with $k = 2$, 4 and 8, respectively using the ADE20k pretrained Segformer model. For each reconstruction, we recorded VRAM usage and frame times for the update steps only; extraction of most-likely class and confidence are excluded. We report these results in Figure 5, where VRAM usage is shown as a percentage of the usage of Naïve Averaging.

CTKH and EF present significant savings in VRAM usage (over 80%) when compared to both RBU and NA and even against a storage optimized Histogram which stores 16-bit integers. Both CTKH and EF exceed real-time rates assuming a 30FPS sensor. EF is slower and uses more memory than CTKH due to the overhead of loading and running its encoder and decoder models, but still achieves significant memory savings.

D. Accuracy and Calibration

We evaluate how compression affects accuracy and calibration using three metrics: mean Intersection-over-Union (mIoU), a measure of accuracy; mean Expected Calibration Error (mECE) a measure of calibration; and Brier score, a combined accuracy and calibration metric.

mIoU is an average of the standard IoU score across classes, which adds more weight to rare classes. If $\hat{V}_c$ are

TABLE I: VRAM usage and agreement to baseline methods in scenes from BS3D ($C = 150$). When a method runs out of memory (OoM), scene completion fraction is reported in parentheses, and agreement is calculated on completed voxels.

Scene	*cafeteria*	*campus*	*central*	*corridor*	*dining*	*foobar*	*hub*	*juice*	*lounge*	*study*	*waiting*
					Max VRAM Usage (Gb) / Scene Completion (%)						
EF-4	3.45	10.20	5.18	3.28	3.69	3.46	3.62	3.16	3.59	3.27	3.15
CTKH-4	2.05	8.62	3.33	1.97	2.35	2.05	2.07	1.74	2.47	1.77	1.74
RBU	13.25	OoM (7.45%)	OoM (38.19%)	13.03	OoM (52.51%)	13.25	13.31	8.06	13.28	8.13	8.04
Histogram	7.75	OoM (16.96%)	OoM (99.23%)	7.77	13.02	7.80	7.86	5.17	7.80	5.21	5.17
					Mean Per-Class Accuracy with Histogram as Pseudo Ground-Truth (%)						
EF-4	98.08	76.21	82.85	100.00	72.06	99.41	99.38	100.00	100.00	100.00	100.00
CTKH-4	99.04	74.66	97.01	99.96	93.41	98.42	100.00	100.00	98.93	100.00	100.00
RBU	99.96	93.23	67.91	100.00	79.25	99.94	99.37	100.00	99.94	100.00	100.00

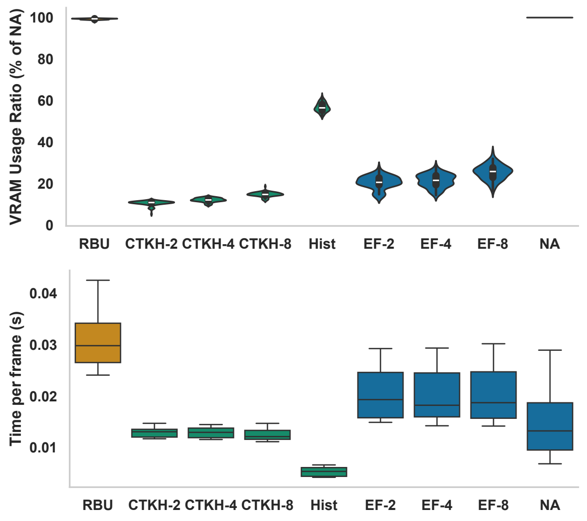


Fig. 5: Memory usage and per-frame update times for semantic fusion methods on 70 ScanNet++ scenes (upper right) with Segformer. Box-and-whisker plots display range, 5th and 95h percentile, and median. Fusion type (averaging, histogram, naïve Bayesian) is indicated by color.

the set of voxels predicted of a certain class and V_c is the ground truth set, then mIoU $= \frac{1}{C}\sum_{c=1}^{C}(\hat{V}_c \cap V_c)/(\hat{V}_c \cup V_c)$.

Likewise, mECE is an average of ECE across classes that is less sensitive to class frequency [13]. It is calculated by binning predicted confidence scores P_v into $B = 10$ bins along equally spaced intervals in the range [0,1]. Let B_{bc} collect all examples i whose confidence scores fall into a bin b and whose ground truth class is c. The accuracy $acc(B_{bc})$ of these examples is the fraction for which the most-likely class $\hat{l}$ matches c. The confidence $conf(B_{bc})$ is the average predicted confidence (i.e., likelihood P_v^c) of examples in B_{bc}. Then, mECE is defined as

$$\text{mECE} = \frac{1}{C}\sum_{c=1}^{C}\sum_{b=1}^{B}\frac{|B_{bc}|}{N_c}|acc(B_{bc}) - conf(B_{bc})| \quad (10)$$

where $N_c = \sum_{b=1}^{B}|B_{bc}|$ is the number of voxels of class c.

Finally, Brier scores are the mean-squared error between the classification indicator and confidence, summed over classes: Brier $= \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}}\sum_{c=1}^{C}(I[c_v^\star = c] - P_v^c)^2$.

Experiments in Figures 6 and 7 compare fusion methods using ESANet and Segformer, respectively, on the ScanNet dataset ($C = 21$). Neither accuracy nor calibration degrade very much with our compression techniques, even for the $k = 1$ case. Compared to their respective baseline techniques, performance degradation of top-k histogram and EF-k is negligible at $k = 4$ and higher, while retaining significant memory savings. Further, we show that these conclusions remain valid for either of the semantic segmentation models used as inputs. We also see that Segformer is better calibrated than ESANet, and EF is more able to leverage confidence at the image-level to outperform CTKH in terms of voxel-level calibration.

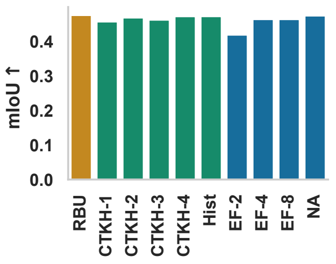
Figure 8 shows our results on the ScanNet++ dataset ($C = 101$) [10]. As was shown on ScanNet, CTKH and EF can closely approximate the results of their respective baseline models (NA and Histogram) as k grows. Note, however, that they reach near parity with $k = 8$. Also note that in such a long tail benchmark, the EF models can better leverage the 2D model's calibration better than CTKH, achieving lower mECEs and Brier scores than its histogram counterparts, as they are agnostic to 2D model confidence.

E. Building-Scale Mapping on BS3D

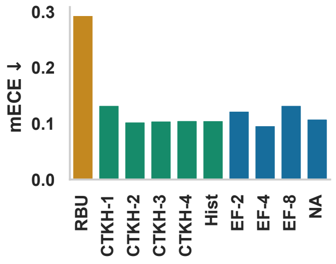
Last, we present results on the BS3D dataset [15] using the ADE20K Segformer pretrained model with 150 classes (C=150). With 5 cm voxel size, we run Histogram, RBU, CTKH-4 and EF-4 on all but one scene from this dataset (RGB-D images are not available for the *lobby* scene at time of writing). As this dataset does not have a semantic ground truth label, we compare the semantic output of the reduced-memory models using high-confidence voxels ($> 80\%$) of Histogram fusion as a pseudo-ground truth, and we report the mean per-class voxel accuracy. For each scene and model combination, we report in Table I the maximum VRAM usage, the percentage of total scene voxels reconstructed, and mean per-class accuracy. RBU is only able to map about a twelfth of the largest *campus* scene, running out of memory at 718,218 out of 9,645,215 occupied voxels. Figure 9 compares the output of RBU and our two methods on BS3D's *campus* scene, with RBU's reconstruction stopped when it runs out of memory. Our compressed approaches map the whole scene. In terms of accuracy, in some scenes, like *central* and *dining*, EF-4 struggles to maintain agreement with CTKH-4 and Histogram, potentially due to the scenes being out of distribution for the trained model, while CTKH-4 remains mostly consistent with Histogram.

V. Conclusion

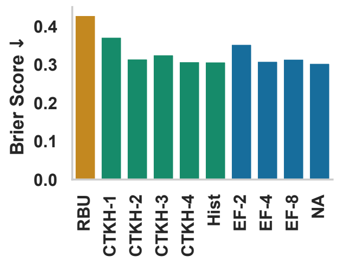
This paper proposed two methods for semantic fusion compression that maintain bounded memory per voxel, and enable scaling to building-scale scenes with millions of



(a) mIoUs

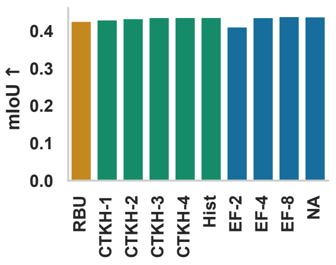


(b) mECEs

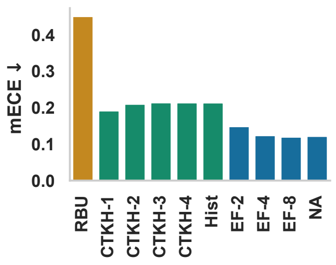


(c) Brier Scores

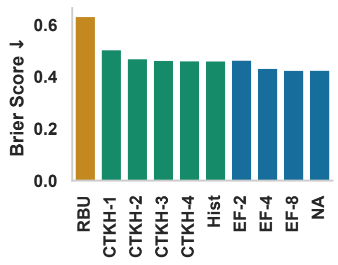
Fig. 6: Accuracy and calibration results on the 20-class dataset ScanNet using the **ESANet** image segmentation model. We show mIoUs (measures classification accuracy, higher is better), mECE (measures calibration, lower is better), and Brier scores (a combined measure, lower is better). Colors indicate underlying fusion method. We see that the memory-efficient reconstruction methods can approximate the performance of the models they approximate, even with small values of k.



(a) mIoUs

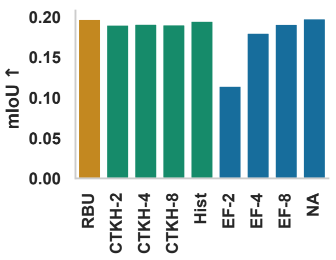


(b) mECEs

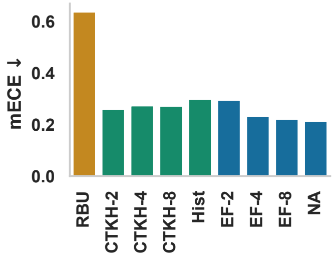


(c) Brier Scores

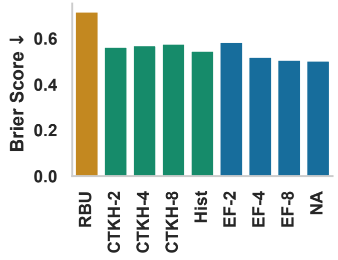
Fig. 7: The same experiments as Fig. 6 using the **Segformer** image segmentation model. Encoded fusion can use better pixel-level calibration to improve voxel-level calibration. Further, we see that the semantic observations from Figure 6 still hold despite different semantic segmentation network types CNN-based and Transformer-based).



(a) mIoUs



(b) mECEs



(c) Brier Scores

Fig. 8: Accuracy and calibration results on the ScanNet++ dataset with 100 semantic classes using the Segformer model. Performance trends from ScanNet persist, though EF2 suffers more heavily in this longer-tail task.

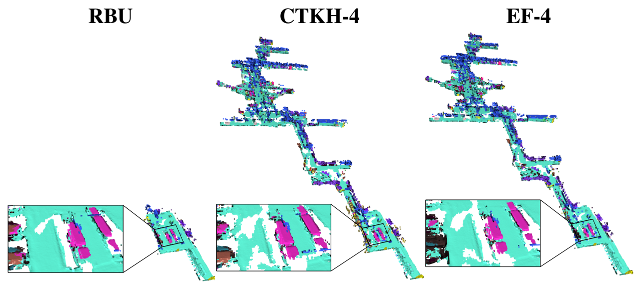


Fig. 9: Semantic reconstruction of the scene *campus* from the BS3D dataset, shown with roof and wall voxels removed. CTKH4 and EF-4 map $\sim 12\times$ more voxels than RBU. Background class masked as transparent.

voxels. With appropriate choices of the k parameter, accuracy and calibration of the fusion pipeline are preserved. CTKH works as a plug-and-play fusion method which can be used without any additional training, but has its performance upper bounded by the calibration and accuracy of histogram fusion. Encoded Fusion, on the other hand, can more closely resemble Naïve Averaging Fusion, which can better benefit from the calibration of the semantic segmentation model it uses as input. It, however, requires additional training to a specific segmentation model, though after pre-training it requires no further tuning at test-time. It can also suffer in cases where the logits it encounters are out of distribution, as was the case in BS3D. We also note that our findings might not hold in cases with fine-grained and highly ambiguous classes, like trying to segment hundreds of plants of the same genus, but different species or hundreds of dog breeds, for instance, being more applicable to class definitions which have less semantic intersection. Future work may investigate other potential approaches, such as dynamic compression. For example, voxels that are old, frequently seen, or whose labels are stable could be compressed more aggressively. We are also interested in deploying our methods on mobile or edge hardware where VRAM is even more limited.

References

[1] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *ICRA*, 2017.

[2] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things," in *IROS*, 2019.

[3] A. Kundu, K. Genova, X. Yin, A. Fathi, C. Pantofaru, L. J. Guibas, A. Tagliasacchi, F. Dellaert, and T. Funkhouser, "Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation," in *CVPR*, Jun. 2022.

[4] A. Hermans, G. Floros, and B. Leibe, "Dense 3D semantic mapping of indoor scenes from RGB-D images," in *ICRA*, 2014.

[5] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, P. Pérez, and P. H. S. Torr, "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *ICRA*, 2015.

[6] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *IJRR*, vol. 40, no. 12-14, 2021.

[7] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader, and L. Carlone, "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *IJRR*, 2024.

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017.

[9] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[10] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, "Scannet++: A high-fidelity dataset of 3d indoor scenes," in *ICCV*, 2023.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *ICML*, vol. 70, 2017.

[12] D. Morilla-Cabello, L. Mur-Labadia, R. Martinez-Cantin, and E. Montijano, "Robust Fusion for Bayesian Semantic Mapping," *arXiv*, Mar. 2023.

[13] J. M. Correia Marques, A. J. Zhai, S. Wang, and K. Hauser, "On the overconfidence problem in semantic 3d mapping," in *ICRA*, 2024, pp. 11 095–11 102.

[14] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *NeurIPS*, vol. 34, 2021.

[15] J. Mustaniemi, J. Kannala, E. Rahtu, L. Liu, and J. Heikkilä, "Bs3d: Building-scale 3d reconstruction from rgb-d images," *arXiv preprint arXiv:2301.01057*, 2023.

[16] L. Li, L. Zhang, Z. Wang, and Y. Shen, "Gs3lam: Gaussian semantic splatting slam," in *ACM Multimedia*, 2024.

[17] A. Asgharivaskasi and N. Atanasov, "Active bayesian multi-class mapping from range and semantic segmentation observations," in *ICRA*, 2021.

[18] M. Runz, M. Buffier, and L. Agapito, "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects," in *International Symposium on Mixed and Augmented Reality*, 2018.

[19] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *IROS*, 2017.

[20] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric Object-Level SLAM," in *3DV*, 2018.

[21] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery," *IEEE RA-L*, 2019.

[22] L. Schmid, J. Delmerico, J. L. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena, "Panoptic Multi-TSDFs: a Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency," in *ICRA*, 2022.

[23] D. Seichter, B. Stephan, S. B. Fischedick, S. Müller, L. Rabes, and H.-M. Gross, "Panopticndt: Efficient and robust panoptic mapping," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7233–7240.

[24] X. Fu, S. Zhang, T. Chen, Y. Lu, L. Zhu, X. Zhou, A. Geiger, and Y. Liao, "Panoptic NeRF: 3D-to-2D Label Transfer for Panoptic Urban Scene Segmentation," *ArXiv*, Mar. 2022.

[25] Y. Xiang and D. Fox, "DA-RNN: Semantic mapping with data associated recurrent neural networks," *arXiv preprint arXiv:1703.03098*, 2017.

[26] M. Qin, W. Li, J. Zhou, H. Wang, and H. Pfister, "Langsplat: 3d language gaussian splatting," in *CVPR*, Jun. 2024.

[27] M. Niessner, M. Zollhofer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM ToG*, vol. 32, no. 6, 2013.

[28] W. Dong, Y. Lao, M. Kaess, and V. Koltun, "Ash: A modern framework for parallel spatial hashing in 3d perception," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, 2023.

[29] S. Bultmann, J. Quenzel, and S. Behnke, "Real-time multi-modal semantic fusion on unmanned aerial vehicles with label propagation for cross-domain adaptation," *Robotics and Autonomous Systems*, vol. 159, 2023.

[30] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[31] T. J. Boerner, S. Deems, T. R. Furlani, S. L. Knuth, and J. Towns, "Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support," in *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, ser. PEARC '23, 2023.

[32] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, 2019.

[33] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H.-M. Gross, "Efficient rgb-d semantic segmentation for indoor scene analysis," in *ICRA*, 2021.