2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Uncertainty from Motion for DNN Monocular Depth Estimation

Soumya Sudhakar, Vivienne Sze, Sertac Karaman

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812222

Abstract— **Deployment of deep neural networks (DNNs) for monocular depth estimation in safety-critical scenarios on resource-constrained platforms requires well-calibrated and efficient uncertainty estimates. However, many popular uncertainty estimation techniques, including state-of-the-art ensembles and popular sampling-based methods, require multiple inferences per input, making them difficult to deploy in latency-constrained or energy-constrained scenarios. We propose a new algorithm, called Uncertainty from Motion (UfM), that requires only one inference per input. UfM exploits the temporal redundancy in video inputs by merging incrementally the per-pixel depth prediction and per-pixel aleatoric uncertainty prediction of points that are seen in multiple views in the video sequence. When UfM is applied to ensembles, we show that UfM can retain the uncertainty quality of ensembles at a fraction of the energy by running only a single ensemble member at each frame and fusing the uncertainty over the sequence of frames. In a set of representative experiments using FCDenseNet and eight in-distribution and out-of-distribution video sequences, UfM offers comparable uncertainty quality to an ensemble of size 10 while consuming only 11.3% of the ensemble's energy and running 6.4× faster on a single Nvidia RTX 2080 Ti GPU, enabling near ensemble uncertainty quality for resource-constrained, real-time scenarios.**

I. INTRODUCTION

Deployment of deep neural networks (DNNs) for monocular depth estimation for resource-constrained robots is an increasingly popular way for sensing the environment [1], [2], [3]. Using a DNN trained to predict per-pixel depth from images enables depth estimation for autonomous vehicles and robotic platforms where a single camera can be used rather than traditional depth sensors that are heavier, larger, and consume more power such as LIDAR, structured light, and stereo cameras. However, in order to deploy in safety-critical scenarios and reason probabilistically to minimize the chance of collisions, we require accurate and efficient estimates of the uncertainty of the DNN prediction.

The total predictive uncertainty of a DNN can come from two sources: uncertainty inherent to the data (*e.g.*, lighting, blur, glare), called aleatoric uncertainty, and uncertainty inherent to the model called epistemic uncertainty [4]. Aleatoric uncertainty stays constant with increased training data, while epistemic uncertainty decreases as the training data increases. Estimating both sources of uncertainty is necessary for capturing the total predictive uncertainty of the DNN. For instance, in Figure 1, DNNs trained on the

Authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139, USA. Emails: {soumyas, sze, sertac}@mit.edu. This work was partially supported by National Science Foundation (NSF) Cyber-Physical Systems program grant no. 1837212, NSF Real-Time Machine Learning program grant no. 1937501, and the MIT-Accenture Fellowship.

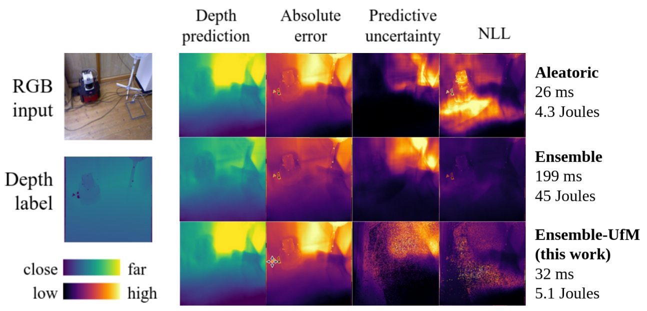

Fig. 1: Uncertainty estimation comparison for an aleatoric network, ensemble, and UfM applied to ensembles on an out-of-distribution cropped example from the TUM RGBD [5] dataset. Lower NLL indicates better uncertainty quality.

NYUDepthV2 dataset [6] predict the depth for an out-of-distribution (OOD) image from the TUM RGBD dataset [5]. This image contains a vacuum cleaner, for which all the DNNs have difficulty estimating the depth as seen by the high error. Aleatoric uncertainty alone fails to predict high uncertainty in the vacuum region. Meanwhile, even though the ensemble also has high error in that region, it correctly predicts higher uncertainty in that region due to its ability to also capture epistemic uncertainty.

We quantify uncertainty by estimating the variance of the DNN prediction. While estimating aleatoric variance is computationally inexpensive [4], estimating epistemic variance is computationally expensive. Techniques used to estimate epistemic variance, such as sampling a stochastic DNN [7] [8] and the state-of-the-art ensemble method [9], [10], involve capturing the variance of predictions using M networks on a single image and require M inferences per input. For example, in Figure 1, although the ensemble returns the best uncertainty quality, it has a latency of 193 ms on a Nvidia RTX 2080 Ti GPU, which makes it far from real-time video speed (33.3 ms for 30 fps). Even for a smaller and faster network such as FastDepth [2], running $M = 10$ inferences per input sequentially would reduce the speed from 178 fps to 17.8 fps on the Nvidia Jetson TX2, reducing the speed of the network to slower than real-time.

The main contribution of this paper is a new algorithm, called Uncertainty from Motion (UfM), in which we consider the variance of predictions across images in a video sequence using a single ensemble member or sampled network on each image. Given the DNN's depth prediction, aleatoric variance prediction, and the pose estimate of the camera, we merge the per-pixel depth predictions and per-pixel aleatoric variance predictions that are multiple views of the same point in 3D space to compute the total predictive variance, which includes epistemic variance, with only one inference per input. UfM is lightweight and works with any DNN that outputs a depth prediction and aleatoric variance prediction or just a depth prediction, without requiring any modifications to the original network architecture or training procedure. For example, when applied to an ensemble on a Nvidia RTX 2080 Ti GPU over eight datasets for the FCDenseNet DNN architecture, UfM obtains on average similar uncertainty quality to an ensemble of size 10 while consuming only 5.0 J at 32 fps compared to the ensemble which consumes 44.5 J at 5 fps.

II. RELATED WORK

DNN uncertainty estimation: Uncertainty estimation for DNNs is a large and active field of research. Nix et al. [11] showed how to learn to predict aleatoric variance through a modified loss function and Kendall et al. [4] extended aleatoric variance estimation for dense prediction tasks. For estimating epistemic variance, sampling-based methods such as Bayesian NNs have been explored, where instead of constant DNN weights, each weight has a distribution that is sampled at inference time to produce a DNN with sampled weights [8] [12] [13]. Gal et al. [7] [4] proposed a popular sampling-based method called MC-Dropout, where dropout is applied at inference time for M inferences and the epistemic variance is the variance of the M predictions. Lakshminarayan et al. [9] proposed the state-of-the-art method of ensembles [14] [10], where M ensemble members that each make depth predictions and aleatoric variance predictions are randomly initialized and trained on randomly ordered datasets; the predictions are combined through a mixture of Gaussians approach to produce the total predictive variance.

Both sampling-based methods and ensembles require M inferences per input. The UfM algorithm proposed in this work is used to reduce the computational cost of sampling-based and ensemble-based methods, requiring only one inference per input while maintaining the uncertainty quality.

Temporal consistency: There exists a rich body of work that has looked at consistency between images for the tasks of structure-from-motion (SfM) [15], [16], [17], self-supervised learning [18], [19], [20], [21], temporally consistent depth prediction [22], [23], and data augmentation [24]. These works use consistency between images to infer structure of the environment or to improve the depth prediction itself. In this work, we use consistency between images to help measure uncertainty of the DNN prediction.

Similar to this work, Liu et al. [25] and Huang et al. [26] consider temporal consistency between images to measure uncertainty of a DNN prediction. In Liu et al. [25], a DNN is trained to take in a sequence of images and predict a non-parametric probability distribution for the depth prediction which is post-processed by two additional DNNs that apply a modified Bayesian filter with learned parameters and increase the resolution respectively. Unlike Liu et al. [25], our work does not introduce any additional DNNs, which reduces computational cost and avoids introducing new unknown sources of DNN uncertainty to estimate the original DNN's uncertainty. Instead, UfM provides a method to accelerate existing DNN ensembles and sampling-based methods without retraining by running a single ensemble member or sampled DNN per image and merging those predictions over time.

The closest work in the spirit of this paper is Huang et al. [26], which also requires only one inference per input by using optical flow to estimate correspondences between images and computing a running average of MC-Dropout class predictions across images for the task of semantic segmentation. While their work can be extended to depth, it cannot handle merging predicted aleatoric variances. Furthermore, an additional DNN is used to compute optical flow which can be computationally expensive and introduces an additional source of unknown DNN uncertainty in order to compute the original DNN's uncertainty. Meanwhile, UfM merges both DNN depth predictions and aleatoric variance predictions through a mixture of Gaussians approach, allowing it to be used to accelerate state-of-the-art ensembles [9] and popular sampling-based aleatoric networks [4]. The method of finding correspondences in UfM does not require an additional DNN, relying instead on a pose estimate, and is lightweight (6.1 ms for 224x224 depth predictions on Nvidia GTX 2080 Ti) such that it can be applied to real-time systems.

III. PROBLEM DEFINITION

Let $\theta_m(\mathbf{X_n}) \rightarrow (\mathbf{Z}_{m,n}, \boldsymbol{\sigma}^2_{m,n})$ be the m'th functional mapping parameterized by weights (DNN) that takes in the n'th image $\mathbf{X_n}$ in video sequence $\mathcal{N} = \{1, ...n, ...N\}$ and outputs per-pixel depth prediction $\mathbf{Z}_{m,n}$ and per-pixel aleatoric variance prediction $\boldsymbol{\sigma}^2_{m,n}$. Let $\mathbf{Y_n}$ be the ground-truth depth label for image $\mathbf{X_n}$ and let $\theta_{1:M}$ be M functional mappings parameterized by weights (*e.g.,* a DNN ensemble of size M or M sampled networks from a BNN). We assume we are given the relative pose ${}^{I_{n-1}}_{I_n}\mathbf{R}$, ${}^{I_{n-1}}_{I_n}\mathbf{t}$ of the camera reference frame I_n at image n with respect to the camera reference frame I_{n-1} at the previous image $n-1$. Since autonomous navigation requires the robot to estimate its pose as part of its localization framework, we believe this assumption is reasonable. It also allows us to incorporate information from other sensor modalities (*e.g.,* IMU). We assume a static world world assumption; dynamic objects can be marked with high uncertainty due to geometric inconsistency between images. We consider the case where the DNNs $\theta_{1:M}$ are fixed, and so we require a solution that does not require retraining which can be expensive and time-consuming. Given DNNs $\theta_{1:M}$ and the pose of the camera, our goal is to the estimate the total predictive uncertainty $\overline{\boldsymbol{\sigma}}^2_n$ for a sequence of images 1 to N while keeping the latency and energy low to enable real-time, energy-efficient uncertainty estimation.

IV. ALGORITHM

We now propose a new algorithm called Uncertainty from Motion (UfM) that merges the depth predictions and aleatoric variance predictions of the DNNs $\theta_{1:M}$ temporally across a sequence of images, allowing us to run only one inference using a single DNN per image. The pseudocode for UfM is presented in Algorithm 1 and our implementation can be found at `https://github.com/mit-lean/ufm`. We next outline the steps of the algorithm at each image.

1) Select and run DNN: Although we have access to all M DNNs $\theta_{1:M}$, the efficiency advantage of UfM comes from only running one inference using a single DNN per image. We cycle through the M networks over the N images such that we run θ_m on the n'th input image $\mathbf{X_n}$ where $m = n$ modulo M to obtain the depth prediction $\mathbf{Z}_{m,n}$ and aleatoric variance prediction $\boldsymbol{\sigma}^2_{m,n}$ (line 2 in Algorithm 1).

2) Project point cloud to image plane: UfM maintains a point cloud C where each point p describes the mean and covariance of the mixture of $K-1$ Gaussians from the $K-1$ views of that point that we have seen before in the previous $n-1$ images. Each point p has a 3D position ${}^{I_{n-1}}\boldsymbol{\mu}_{1:K-1}$ w.r.t. the camera reference frame in frame $n-1$, a covariance matrix ${}^{I_{n-1}}\boldsymbol{\Sigma}_{1:K-1}$ w.r.t. the camera reference frame in frame $n-1$, and a counter for the number of times this point has been seen, here equal to $K-1$. For each point in the point cloud, we rotate the position and covariance matrix from the previous camera's reference frame to the current camera's reference frame using

$$\begin{aligned}{}^{I_n}\boldsymbol{\mu}_{1:K-1} &= {}^{I_{n-1}}_{I_n}\mathbf{R}\,{}^{I_{n-1}}\boldsymbol{\mu}_{1:K-1} + {}^{I_{n-1}}_{I_n}\mathbf{t},\\ {}^{I_n}\boldsymbol{\Sigma}_{1:K-1} &= {}^{I_{n-1}}_{I_n}\mathbf{R}\,{}^{I_{n-1}}\boldsymbol{\Sigma}_{1:K-1}\,{}^{I_{n-1}}_{I_n}\mathbf{R}^T,\end{aligned} \quad (1)$$

(lines 5-6 in Algorithm 1). We drop the reference frame notation for rest of this section for brevity. Next, to find correspondences between the point cloud, the depth prediction $\mathbf{Z_{m,n}}$, and the aleatoric variance prediction $\boldsymbol{\sigma^2_{mn}}$, we project each point in the point cloud back to the image plane and calculate the pixel it projects to (line 7 in Algorithm 1), using

$$\begin{aligned}u' &= \lfloor f_x\mu_{1:K-1,x}/\mu_{1:K-1,z} + c_x\rceil\\ v' &= \lfloor f_y\mu_{1:K-1,y}/\mu_{1:K-1,z} + c_y\rceil,\end{aligned} \quad (2)$$

where $\mu_{1:K-1,x}$, $\mu_{1:K-1,y}$, $\mu_{1:K-1,z}$ are the x, y, z components of $\boldsymbol{\mu}_{1:K-1}$, and f_x, f_y, c_x, and c_y are the camera focal lengths and image center from the intrinsic matrix F. The point $\boldsymbol{\mu}_{1:K-1}$ is within bounds of the new image if it satisfies

$$u' < u_{max} \;\wedge\; u' \geq 0 \;\wedge\; v' < v_{max} \;\wedge\; v' \geq 0, \quad (3)$$

where u_{max} and v_{max} are the width and height of the image. If pixel u', v' is in bounds of the new image, we interpret the depth prediction $Z_{m,n}$ at pixel u', v' and aleatoric variance prediction $\sigma^2_{m,n}$ at pixel u', v', as the K'th view of $\boldsymbol{\mu}_{1:K-1}$ ($K > 1$). If pixel u', v' is out of bounds of the new image, we interpret $Z_{m,n}$ and $\sigma^2_{m,n}$ to be the first view of a new point we have not seen before ($K = 1$). If u', v' is within bounds of the image, we update a mask of the image D indexed at pixel u', v' to be true and store a pointer to the point $\boldsymbol{\mu}_{1:K-1}$ (lines 8-9 in Algorithm 1). We repeat this process for all points in the point cloud.

3) Project image to 3D space: Let u, v index all pixels in the image. For each $Z_{m,n}$ and $\sigma^2_{m,n}$ at pixel u, v, we project $Z_{m,n}$ and $\sigma^2_{m,n}$ to 3D space (line 11 in Algorithm 1) using

$$\boldsymbol{\mu}_K = [\tfrac{u-c_x}{f_x}Z_{m,n} \quad \tfrac{v-c_y}{f_y}Z_{m,n} \quad Z_{m,n}]^T$$
$$\boldsymbol{\Sigma}_K = \begin{bmatrix} \left(\frac{u-c_x}{f_x}\right)^2\sigma^2_{m,n} & \frac{u-c_x}{f_x}\frac{v-c_y}{f_y}\sigma^2_{m,n} & \frac{u-c_x}{f_x}\sigma^2_{m,n}\\ \frac{u-c_x}{f_x}\frac{v-c_y}{f_y}\sigma^2_{m,n} & \left(\frac{v-c_y}{f_y}\right)^2\sigma^2_{m,n} & \frac{v-c_y}{f_y}\sigma^2_{m,n}\\ \frac{u-c_x}{f_x}\sigma^2_{m,n} & \frac{v-c_y}{f_y}\sigma^2_{m,n} & \sigma^2_{m,n}\end{bmatrix}. \quad (4)$$

4) Update point cloud: If mask D indexed at pixel u, v is true, then pixel u, v represents the K'th view of a point we have seen before ($K > 1$), and we update $\boldsymbol{\mu}_{1:K-1}$ and $\boldsymbol{\Sigma}_{1:K-1}$ with the new K'th Gaussian with mean $\boldsymbol{\mu}_K$ and covariance $\boldsymbol{\Sigma}_K$ (lines 12-14 in Algorithm 1). To merge the K'th measurement $\boldsymbol{\mu}_K$ and $\boldsymbol{\Sigma}_K$ into $\boldsymbol{\mu}_{1:K-1}$ and $\boldsymbol{\Sigma}_{1:K-1}$, we assume a uniformly-weighted Gaussian mixture model $K^{-1}\sum_k^K \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ in which the k'th measurement of the point contributes a Gaussian parameterized by $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. We approximate the distribution of the mixture as a Gaussian parameterized by the mean and covariance of the mixture. The mean of the mixture model is

$$\boldsymbol{\mu}_{1:K} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\mu}_k, \quad (5)$$

and the covariance matrix of the mixture model is

$$\boldsymbol{\Sigma}_{1:K} = \frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{\Sigma}_k + (\boldsymbol{\mu}_k - \boldsymbol{\mu}_{1:K})(\boldsymbol{\mu}_k - \boldsymbol{\mu}_{1:K})^T). \quad (6)$$

We incrementally update the mean and covariance of the mixture to avoid having to store the depth prediction and aleatoric variance predictions from previous images. As each point in 3D space is seen for the K'th time, its new Gaussian is added to the mixture of $K-1$ Gaussians for that point. The incremental update to the mixture mean is

$$\boldsymbol{\mu}_{1:K} = \boldsymbol{\mu}_{1:K-1} + (\boldsymbol{\mu}_K - \boldsymbol{\mu}_{1:K-1})/K, \quad (7)$$

and the incremental update to the mixture variance is

$$\begin{aligned}\boldsymbol{\Sigma}_{1:K} =& \frac{K-1}{K}(\boldsymbol{\Sigma}_{1:K-1} + \boldsymbol{\mu}_{1:K-1}\boldsymbol{\mu}^T_{1:K-1}) - \boldsymbol{\mu}_{1:K}\boldsymbol{\mu}^T_{1:K}\\ &+ \frac{1}{K}(\boldsymbol{\Sigma}_K + \boldsymbol{\mu}_K).\end{aligned} \quad (8)$$

If mask D indexed at pixel u, v is false, then none of the points in the point cloud projected to pixel u, v and it represents the first view of a new point ($K = 1$). We initialize a new point in the point cloud with mean $\boldsymbol{\mu}_1$ and covariance $\boldsymbol{\Sigma}_1$ from Eq. 4 with the counter $K = 1$ (lines 15-16 in Algorithm 1).

The total predictive uncertainty $\overline{\sigma^2_n}$ for the pixel u, v is the z-variance component of $\boldsymbol{\Sigma}_{1:K}$ of the updated or added point (line 17 in Algorithm 1). We repeat this process for all the pixels in the predicted depth map, and obtain the total predictive uncertainty $\overline{\boldsymbol{\sigma^2_n}}$ for all pixels in the n'th image.

5) Threshold point cloud: The time complexity of UfM per image is $\mathcal{O}(u_{max}v_{max} + |C|)$ where $u_{max}v_{max}$ is the number of pixels in the depth prediction and $|C|$ is the

Algorithm 1: Uncertainty from Motion (UfM)

```
Input  : RGB input $\mathbf{X_n}$, relative pose ${}^{I_{n-1}}_{I_n}\mathbf{R}$, ${}^{I_{n-1}}_{I_n}\mathbf{t}$ for
         $n$'th image $\in [1, N]$, DNNs $\theta_{1:M}$, camera
         intrinsic matrix $F$, image dimensions $u_{max}, v_{max}$
Output: Depth prediction $\mathbf{Z}_{m,n}$, total predictive variance
         $\overline{\boldsymbol{\sigma}^2_n}$ for $n$'th image $\in [1, N]$
1  for $n \leftarrow 1$ to $N$ do
2      $\mathbf{Z}_{m,n}, \boldsymbol{\sigma}^2_{m,n} \leftarrow$ select_and_run_dnn($n, M$)
3      $\mathbf{D} \leftarrow \mathbf{0}$
4      for $p \in C$ do
5          $\boldsymbol{\mu}_{1:K-1} \leftarrow$ rotate_pos(${}^{I_{n-1}}_{I_n}\mathbf{R}$, ${}^{I_{n-1}}_{I_n}\mathbf{t}$) (Eq. 1)
6          $\boldsymbol{\Sigma}_{1:K-1} \leftarrow$ rotate_cov(${}^{I_{n-1}}_{I_n}\mathbf{R}$, ${}^{I_{n-1}}_{I_n}\mathbf{t}$) (Eq. 1)
7          $u', v' \leftarrow$ project_to_image($\boldsymbol{\mu}_{1:K-1}, F$) (Eq. 2)
8          if in_bounds($u', v', u_{max}, v_{max}$) (Eq. 3) then
9              $\mathbf{D}[u', v'] \leftarrow 1$
10     for $u \leftarrow 0$ to $u_{max}, v \leftarrow 0$ to $v_{max}$ do
11         $\boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K \leftarrow$ project_to_3D($Z_{m,n}, \sigma^2_{m,n}, u, v, F$)
             (Eq. 4)
12         if $\mathbf{D}[u, v] = 1$ then
13             $\boldsymbol{\mu}_{1:K} \leftarrow$ update_mean($\boldsymbol{\mu}_K$, $\boldsymbol{\mu}_{1:K-1}, K$)
                 (Eq. 7)
14             $\boldsymbol{\Sigma}_{1:K} \leftarrow$ update_cov
                 ($\boldsymbol{\Sigma}_{1:K-1}, \boldsymbol{\Sigma}_K, \boldsymbol{\mu}_{1:K}, \boldsymbol{\mu}_{1:K-1}, \boldsymbol{\mu}_K, K$)
                 (Eq. 8)
15         else
16             $C \leftarrow C \cup (\boldsymbol{\mu}_K, \boldsymbol{\Sigma}_K, 1)$
17         $\overline{\boldsymbol{\sigma}^2_n}[u, v] \leftarrow \boldsymbol{\Sigma}_{1:K}[2, 2]$
18     if $|C| > C_{max}$ then
19         $C \leftarrow$ threshold_point_cloud($C$)
```

number of points in the point cloud C. While the image dimension is generally fixed for a task, $|C|$ will keep growing if left unbounded. For efficiency, we apply a threshold C_{max} to the point cloud so that when $|C| > C_{max}$, points are removed from the point cloud (lines 18-19 in Algorithm 1). We use the rule that points out of view are removed first, followed by oldest points. Note, thresholding can cause the total predictive variance to be discontinuous across images when a point is thrown away and then seen again.

Note, we present operations per point in the point cloud and per pixel in the image in the pseudocode for simplicity; our implementation uses vectorized matrix operations to speed the processing on the GPU. Note, in this work, UfM is only applied to depth estimation. However, once the correspondences between frames are calculated, the same correspondences can be used to apply UfM to other dense prediction tasks such as semantic segmentation.

V. Metrics for Uncertainty Quality

While there is no ground-truth for uncertainty, we would like the total predictive variance to behave such that it is high when the accuracy is low, and it is low when the accuracy is high. The pixel-wise negative-log likelihood (NLL) captures this trend when it is minimized [27], given for the Gaussian case by

$$NLL = \frac{1}{2\overline{\sigma^2_n}}(Y_n - Z_{m,n})^2 + \frac{1}{2}ln(2\pi\overline{\sigma^2_n}). \quad (9)$$

We show the NLL for each pixel in Figure 1, where the proposed method has a low NLL like the ensemble. We can also average the pixel-wise NLL across all pixels across all images in a sequence to obtain an average NLL.

In addition to NLL, calibration curves are widely used to quantify uncertainty quality and they can indicate when the total predictive variance is underestimating or overestimating its uncertainty [28], [27], [4], [9]. We first look at the calibration of accuracy vs. confidence. The confidence of the depth prediction is the probability that the true depth lies within in an interval δ around the depth prediction (*e.g.,* δ = $\pm 25\%$ of depth prediction) and can be calculated from the depth prediction and total predictive variance. For each pixel u, v, we calculate the probability mass within interval δ for the Gaussian distribution parameterized by the depth prediction and total predictive variance $\mathcal{N}(Z_{m,n}, \overline{\sigma^2_n})$. If an uncertainty estimate is perfectly calibrated, there is a one-to-one correspondence between the confidence of the pixels and the average accuracy of the pixels with that confidence level. For example, for all pixels that the model is 60% confident in, the average accuracy of those pixels should be 60%. If the accuracy is instead 40%, the model is overconfident, and if the accuracy is instead 80%, the model is under-confident. The expected calibration error (ECE_δ) is the weighted RMSE of the confidence vs. accuracy calibration from perfect calibration, given by

$$ECE_\delta = \sum_b^B \frac{|b|}{S} |acc(b) - conf(b)| \quad (10)$$

where $|b|$ is the number of pixels in confidence bin b of B total bins, and $S = Nu_{max}v_{max}$ which is the total number of pixels in the sequence [28] [27].

We also consider the calibration curve of expected frequency vs. observed frequency used in prior works [4], [29], where for a range of probabilities, we calculate the inverse CDF of $\mathcal{N}(Z_{m,n}, \overline{\sigma^2_n})$ over a range of probabilities for each pixel and consider observed frequency to be the fraction of corresponding ground-truth pixels that fall under the inverse CDF. Under perfect calibration, the expected and observed frequency are equivalent, and we can calculate

$$ECE_{freq} = \sum_b^B \frac{|b|}{S} \left|exp_{freq}(b) - obs_{freq}(b)\right|. \quad (11)$$

Uncertainty quality for an uncertainty estimation method is higher when ECE_δ, ECE_{freq} and NLL are lower.

VI. Experimental Setup

In this section, we discuss the experimental set-up for the results in Section VII. We use a threshold $C_{max} = 100,000$ points and we test two use cases of UfM: Ensemble-UfM where $\theta_{1:M}$ is an ensemble with M ensemble members as in Lakshminarayan et al. [9] and MC-Dropout-UfM where $\theta_{1:M}$

is an aleatoric network that is sampled M times with different dropouts as in Kendall et al. [4]. We test three architectures: 1) FC-DenseNet [30], 2) ResNet50-UpProj [31], and 3) FastDepth [2] and train an ensemble of size $M = 10$ aleatoric networks, a MC-Dropout network with dropout probability $p = 0.2$, and a network that predicts only depth with l_1 loss for each architecture. Note, ResNet50-UpProj and FastDepth both have ImageNet pretrained encoder weights which are not randomly initialized for the ensemble. All networks are trained for 50 epochs on the NYUDepthV2 dataset [6] with the SGD optimizer.

ORBSLAM2 [32], [33] is used to obtain pose estimates for the sequences where ground-truth pose is unavailable. All RGB inputs are processed into 224x224x3 inputs as in [2]. We run all the experiments in PyTorch on a Nvidia RTX 2080 Ti GPU. We measure the average power P the GPU consumes during the experiment; we calculate energy using $E = Pt$.

VII. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of UfM with different methods, network architectures, datasets, and pose estimates. Table I shows the validation δ_1 accuracy on the official NYUDepthV2 validation set, where δ_i = % of $\boldsymbol{Z_{m,n}}$ s.t. $\frac{|Z_{m,n}-Y_n|}{Y_n} < (1.25^i - 1)$, where δ_1 accuracy is the fraction of depth predictions $Z_{m,n}$ that are within 25% of ground-truth Y_n. Note, many prior works use $\overline{\delta_i}$ = % of $\boldsymbol{Z_{m,n}}$ s.t. $\max(\frac{Z_{m,n}}{Y_n}, \frac{Y_n}{Z_{m,n}}) < 1.25^i$ to measure relative error [34] [35] [36] [2] [3]. However, this expression does not evaluate to relative error when $Y_n > Z_{m,n}$; we report $\overline{\delta_i}$ in Table I for comparison to previous works, and use δ_1 for the remainder of the results. We see that the accuracy on the validation set is similar to that in the literature [30], [31], [2].

TABLE I: δ_1, $\overline{\delta_1}$ accuracy on NYUDepthv2 official val. set

Method	FCDenseNet	ResNet50_UpProj	FastDepth
Depth only	81.0, 77.0	84.1, 80.7	81.7, 77.5
Aleatoric	80.7, 77.2	83.6, 79.9	80.7, 76.7
MC-Dropout	78.1, 73.5	82.1, 78.3	78.2, 74.3
Ensemble	81.0, 77.0	83.1, 80.0	80.7, 76.5

TABLE II: Numbering for sequences and datasets used

Num.	Sequence	Num.	Sequence
1	TUM: long office	5	KITTI: 0022
2	TUM: pioneer slam3	6	KITTI: 0061
3	TUM: room	7	NYU: bathroom 0003
4	KITTI: 0009	8	NYU: living room 0003

For the remainder of the results, we test on video sequences where UfM can be applied. We test on TUM RGBD (indoor out-of-distribution) [5], KITTI (outdoor out-of-distribution) [37], and NYUDepthV2 (indoor in-distribution) [6], numbered in Table II for reference. Since there is a lack of uncertainty quality baselines to compare

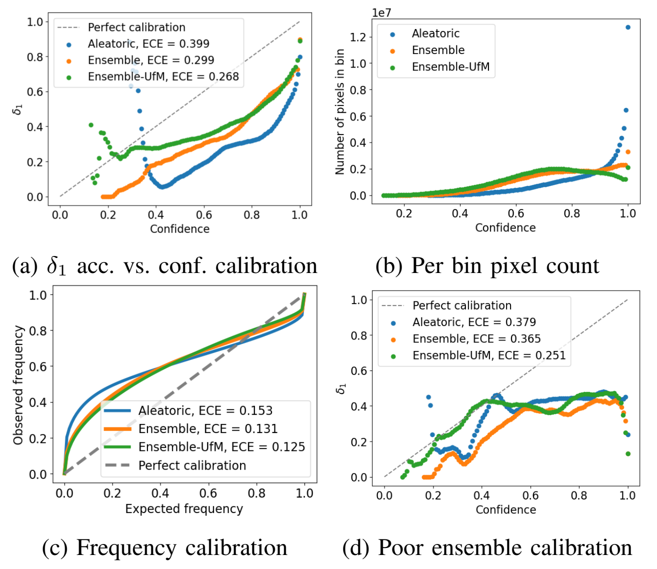

Fig. 2: Calibration curves (a,b,c) Sequence 1 (d) Sequence 2

to in the literature for video sequences, we evaluate the UfM algorithms in relation to an aleatoric only network (one inference per image) [4], an aleatoric MC-Dropout network (10 inferences per image) [4], and an ensemble of size $M = 10$ [9] (10 inferences per image) as baselines.

TABLE III: UfM applied with FC-DenseNet on Seq. 1 with ground-truth pose; arrows show favorable direction for metric

Method	$\delta_1 \uparrow$	$ECE_{\delta_1} \downarrow$	NLL $\downarrow$	t (ms) $\downarrow$
Aleatoric	0.471	0.399	8.20	**25.9**
MC-Dropout	0.460	0.265	3.86	194.8
MC-Dropout-UfM	0.454	**0.214**	**3.35**	33.2
Ensemble	**0.475**	0.299	4.63	199.1
Ensemble-UfM	0.458	0.269	4.35	32.5
Depth only-UfM	0.451	0.508	6652.9	31.7
Full MC-Dropout-UfM	0.460	0.216	3.87	200.7
Full ensemble-UfM	0.476	0.233	3.87	201.5

The top of Table III shows the results for MC-Dropout-UfM, Ensemble-UfM, and the baselines. We see that MC-Dropout-UfM and Ensemble-UfM both outperform the network that outputs aleatoric variance only. Furthermore, MC-Dropout-UfM is able to maintain similar uncertainty quality to MC-Dropout at a fraction of the latency; this trend also holds true for Ensemble-UfM. UfM adds an average latency of 6.1 ms, showing it has lightweight overhead.

Although UfM is motivated by the need to speed up ensembles and sampling based networks, it can be applied to any network that produces a depth prediction. On the bottom of Table III (italicized), we see that UfM applied to a network that does depth prediction only ($\boldsymbol{\sigma}^2_{m,n} = \mathbf{0}$) performs very poorly indicating the importance of incorporating aleatoric variance predictions. In addition, UfM can also be applied in the case where we run the full $\theta_{1:M}$ networks on each image, but we lose the advantage in efficiency.

Table IV and Table V show a subset of results for different architectures and datasets for Ensemble-UfM and the base-

TABLE IV: Results of Ensemble-UfM vs. baselines on three architectures on Sequence 2

Arch.	Method	$\delta_1 \uparrow$	$ECE_{\delta_1} \downarrow$	$\delta_2 \uparrow$	$ECE_{\delta_2} \downarrow$	$ECE_{freq} \downarrow$	NLL $\downarrow$	t (ms) $\downarrow$	P [W]	E [J] $\downarrow$
FC-DN	Aleatoric	**0.418**	0.379	**0.766**	0.213	0.122	8.58	**25.7**	163.4	**4.20**
	Ensemble	0.356	0.365	0.712	0.246	0.138	5.07	195.1	232.7	45.84
	Ensemble-UfM	0.399	**0.251**	0.731	**0.190**	**0.101**	**4.32**	30.3	165.1	5.01
ResNet50	Aleatoric	0.320	0.554	0.585	0.388	0.249	39.54	**16.4**	191.0	**3.13**
	Ensemble	0.318	0.512	0.586	0.384	0.286	10.93	148.0	238.2	35.25
	Ensemble-UfM	**0.326**	**0.413**	**0.607**	**0.328**	**0.205**	**7.99**	24.2	178.5	4.32
FastDepth	Aleatoric	**0.474**	0.383	**0.793**	**0.189**	0.129	14.34	**6.0**	64.0	**0.38**
	Ensemble	0.412	0.342	0.663	0.306	0.184	6.83	49.9	69.4	3.46
	Ensemble-UfM	0.423	**0.240**	0.727	0.203	**0.101**	**5.08**	13.6	68.8	0.93

TABLE V: Results of Ensemble-UfM vs. baselines on FCDenseNet architecture and different sequences

Seq.	Method	$\delta_1 \uparrow$	$ECE_{\delta_1} \downarrow$	$\delta_2 \uparrow$	$ECE_{\delta_2} \downarrow$	$ECE_{freq} \downarrow$	NLL $\downarrow$	t (ms) $\downarrow$	P [W]	E [J] $\downarrow$
3	Aleatoric	0.358	0.495	0.651	0.339	0.301	6.99	**26.0**	171.6	**4.46**
	Ensemble	**0.390**	0.357	**0.676**	0.292	0.253	4.28	196.3	233.1	45.76
	Ensemble-UfM	0.366	**0.305**	0.656	**0.282**	**0.243**	**3.55**	30.8	167.3	5.15
4	Aleatoric	0.000	0.700	**0.009**	0.948	**0.490**	53.85	**25.3**	163.8	**4.14**
	Ensemble	0.000	0.606	0.004	0.920	0.570	33.38	197.3	219.5	43.31
	Ensemble-UfM	0.000	**0.539**	0.006	**0.868**	**0.490**	**28.57**	32.5	161.6	5.25
5	Aleatoric	0.000	0.697	**0.019**	0.942	**0.570**	58.10	**25.7**	168.3	**4.33**
	Ensemble	0.000	0.566	0.014	0.881	0.490	36.47	197.9	227.3	44.98
	Ensemble-UfM	0.000	**0.546**	**0.019**	**0.858**	**0.570**	**32.67**	30.5	158.3	4.83
6	Aleatoric	**0.001**	0.751	**0.031**	0.947	0.490	56.68	**25.6**	165.8	**4.24**
	Ensemble	**0.001**	0.600	0.020	0.898	0.490	34.10	199.9	232.5	46.48
	Ensemble-UfM	**0.001**	**0.557**	0.026	**0.862**	**0.489**	**29.58**	30.7	160.7	4.93
7	Aleatoric	0.199	0.702	**0.637**	**0.356**	0.441	3.11	**24.1**	161.5	**3.89**
	Ensemble	**0.240**	0.604	0.576	0.410	**0.404**	2.40	198.5	193.4	38.39
	Ensemble-UfM	0.216	**0.553**	0.605	0.364	0.428	**1.75**	30.1	160.0	4.80
8	Aleatoric	**0.723**	0.211	**0.964**	0.033	0.234	1.20	**24.7**	164.3	**4.06**
	Ensemble	0.689	0.224	**0.964**	**0.032**	0.252	1.24	196.4	235.0	46.15
	Ensemble-UfM	0.707	**0.143**	**0.964**	0.034	**0.227**	**0.98**	30.7	156.1	4.79

lines. We see that Ensemble-UfM obtains close to ensemble uncertainty quality and in some cases, surpasses ensemble uncertainty quality at a much lower latency and energy. For FC-DenseNet, Ensemble-UfM consumes on average 11.3% of the energy that an ensemble consumes and runs 6.4× faster. On all three architectures, Ensemble-UfM can run at or faster than real-time ($<$ 33.3 ms for 30 fps) with comparable uncertainty quality to ensembles. These savings will increase when compared to larger ensembles. Note, all the networks perform poorly both in accuracy and uncertainty quality on Sequences 4-6 though Ensemble-UfM still returns the lowest NLL; this behavior is likely due to the KITTI dataset being extremely out-of-distribution (outdoor vs. indoor).

We next look at calibration curves in Figure 2. Ensemble-UfM performs close the ensemble baseline and is less over-confident than the aleatoric network. There are high accuracy pixels that are classified as low confidence in Ensemble-UfM in Figure 2a, but as seen in Figure 2b, there are few pixels in those bins, lessening the effect on ECE by the poorly calibrated bins. Ensemble-UfM can help mimic ensembles, but if ensemble uncertainty quality is itself poor, Ensemble-UfM will mimic the poor quality as in Figure 2d.

In Table VI, we study the effect of noise in the pose estimate. We see that UfM with a noisy pose returns similar uncertainty quality to UfM with a ground-truth pose. At times, UfM with a noisy pose seems to perform better such as in Sequence 3; this behavior may be due uncertainty being underestimated by Ensemble-UfM, such that when a noisy pose increases the geometric inconsistency and therefore, the uncertainty, the metrics improve. It is possible odometry with just an IMU may suffice in the future since we do not need to consider drift over a long trajectory when using a threshold C_{max} as UfM will maintain a local point cloud.

TABLE VI: Effect of different VO modalities on uncertainty

Seq.	Pose	Traj. RMSE	$ECE_{\delta_1} \downarrow$	$ECE_{freq} \downarrow$	NLL $\downarrow$
1	GT	0	0.269	0.125	4.34
	RGBD	0.010	0.251	0.121	4.07
	Mono.	1.22	**0.223**	**0.114**	**3.82**
2	GT	0	**0.251**	**0.101**	4.32
	RGBD	0.049	0.399	**0.101**	**4.04**
3	GT	0	0.366	0.243	3.55
	RGBD	0.05	0.304	0.244	3.54
	Mono.	0.289	**0.248**	**0.220**	**2.77**

VIII. CONCLUSION

In this paper, we propose Uncertainty from Motion (UfM), an algorithm that enables us to merge depth predictions and aleatoric variance predictions over a sequence of images to speed up ensembles and sampling-based methods for uncertainty estimation. UfM retains the methods' uncertainty quality while running only one inference per input, making high quality uncertainty a possibility for real-time, resource-constrained scenarios.

References

[1] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia, "Learning monocular depth estimation infusing traditional stereo knowledge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9799–9809.

[2] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6101–6108.

[3] M. Song, S. Lim, and W. Kim, "Monocular depth estimation using laplacian pyramid-based depth residuals," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[4] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *arXiv preprint arXiv:1703.04977*, 2017.

[5] 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.

[6] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[7] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[8] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1613–1622.

[9] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *arXiv preprint arXiv:1612.01474*, 2016.

[10] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *arXiv preprint arXiv:1906.02530*, 2019.

[11] D. A. Nix and A. S. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, vol. 1. IEEE, 1994, pp. 55–60.

[12] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson, "Cyclical stochastic gradient mcmc for bayesian deep learning," *arXiv preprint arXiv:1902.03932*, 2019.

[13] J. Heek and N. Kalchbrenner, "Bayesian inference for large scale image classification," *arXiv preprint arXiv:1908.03491*, 2019.

[14] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 8803–8812.

[15] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[16] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International journal of computer vision*, vol. 9, no. 2, pp. 137–154, 1992.

[17] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[18] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1851–1858.

[19] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3288–3295.

[20] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova, "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8001–8008.

[21] J.-W. Bian, H. Zhan, N. Wang, Z. Li, L. Zhang, C. Shen, M.-M. Cheng, and I. Reid, "Unsupervised scale-consistent depth learning from video," *International Journal of Computer Vision*, vol. 129, no. 9, pp. 2548–2564, 2021.

[22] X. Luo, J.-B. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 71–1, 2020.

[23] A. Duzceker, S. Galliani, C. Vogel, P. Speciale, M. Dusmanu, and M. Pollefeys, "Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 324–15 333.

[24] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8856–8865.

[25] C. Liu, J. Gu, K. Kim, S. G. Narasimhan, and J. Kautz, "Neural rgb (r) d sensing: Depth and uncertainty from a video camera," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 986–10 995.

[26] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun, "Efficient uncertainty estimation for semantic segmentation in videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 520–535.

[27] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[28] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[29] A. Amini, W. Schwarting, A. Soleimany, and D. Rus, "Deep evidential regression," *arXiv preprint arXiv:1910.02600*, 2019.

[30] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 11–19.

[31] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[32] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[33] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[34] L. Ladicky, J. Shi, and M. Pollefeys, "Pulling things out of perspective," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 89–96.

[35] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *arXiv preprint arXiv:1406.2283*, 2014.

[36] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4796–4803.

[37] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.