2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

RMap: Millimeter-Wave Radar Mapping Through Volumetric Upsampling

Ajay Narasimha Mopidevi, Kyle Harlow, and Christoffer Heckman[1]

***Abstract*— Millimeter Wave Radar is being adopted as a viable alternative to lidar and radar in adverse visually degraded conditions, such as in the presence of fog and dust. However, this sensor modality suffers from severe sparsity and noise under nominal conditions, which makes it difficult to use in precise applications such as mapping. This work presents a novel solution to generate accurate 3D maps from sparse radar point clouds. RMap uses a generative transformer architecture which upsamples, denoises, and fills the incomplete radar maps to resemble lidar maps. We test this method on the ColoRadar dataset to demonstrate its efficacy.**

I. Introduction

Many recent advancements in navigation use lidar or camera sensors for odometry estimation and mapping [1]. While these sensors provide highly reliable data under normal conditions, they are severely affected in visually degraded environments (VDEs) such as fog, smoke, rain, and snow. In VDEs, lidars can provide false returns, requiring heavy filtering for mitigation [2], and in cameras, the geometry used to track features can be obscured [3].

Millimeter-wave (mmWave) radar sensors are a compelling alternative in adversarial conditions as their longer wave lengths allow measurements to bypass particulate clutter [4]. Common automotive-grade mmWave radar operates between 76-81 GHz and can provide centimeter-level accuracy signal detections.

However, mmWave radar suffers from several key drawbacks compared to other ranging sensors such as lidar. Radar is susceptible to various types of noise, resulting from constructive and destructive interference from the projected wave patterns reflecting off the environment. These wave patterns can lead to spurious returns, lost returns, and multi-path reflections [5]. Additionally, radar data is often sparse, especially when examining the constant false alarm, non-maximal suppression filtered targets. For example, automotive-grade imaging radar typically produces a maximum of just a few hundred points per scan [6]. Furthermore, radar systems generally have significantly lower resolution compared to competing lidar systems.

Traditionally, radar applications have been limited to collision avoidance for autonomous vehicles, rather than high-precision tasks such as odometry estimation and mapping. However, recent research has led to the development of robust systems capable of generating odometry and filtering radar data for mapping purposes [7]–[9]. We present one such solution, RMap (RadarMapping), a method to generate high-precision 3D maps using radar point clouds extracted from an mmWave sensor. We present an end-to-end pipeline for generating the 3D maps from radar point clouds and demonstrate how these maps can be leveraged to construct a 3D map resembling lidar-based maps through deep learning techniques. Furthermore, we show examples comparing the original radar maps, lidar maps, and the map generated on radar data using our method in Figure 3.

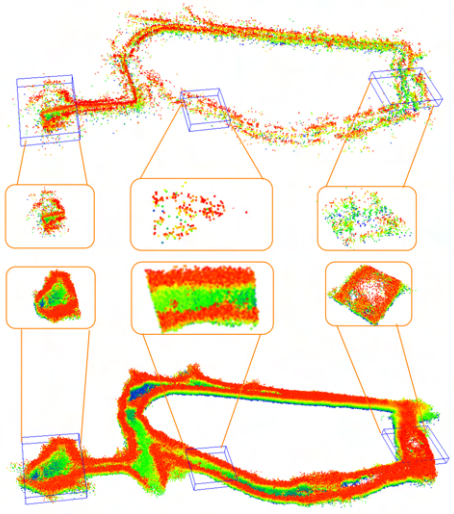

Fig. 1: A qualitative comparison between a map generated using just the Radar Octomap method [3], and RMap, our novel upsampling method. The top row represents the initial radar map generated with Radar Octomap, and the bottom row represents the map generated using RMap. The patches show our method filtering noise from regions of free space, as well as upsampling and filling sparse and empty regions respectively.

Our contributions to the state-of-the-art include:

- We have developed a farthest point sampling (FPS) based patch generation algorithm, which effectively represents the 3D maps as partial point clouds.
- We introduce UpPoinTr, a generative transformer architecture, that performs upsampling, denoising, and fills the incomplete maps.
- Utilizing the radar sensor model for octomap [3], [10], we have generated 3D radar maps for the ColoRadar dataset and processed patches of these maps through UpPoinTr to closely resemble lidar-based maps after reassembly.

[1]All authors are with the Department of Computer Science, University of Colorado Boulder, Boulder CO, USA christoffer.heckman@colorado.edu

- Our work represents the first known application of point cloud completion networks to radar-based 3D scene generation tasks.

II. Background

As radar is still a new sensor to robotics we provide a primer on frequency-modulate continuous-wave (FMCW) radar and then analyze related works to both mapping with radar along with current architectures for point completion.

A. Radar Primer

FMCW radar operates by transmitting a sequence of chirps, or waves with linearly increasing frequency from a series of antennae. These signals are reflected back from the environment and then captured by another series of receive antennae. A mixer subtracts the transmitted signal from the received signal, generating an intermediate frequency correlated along individual chirps, across chirps, and by each transmit and receive antenna. The series of digital samples from an analog-to-digital converter resulting in these measurements is referred to as a data-cube [6].

After assembling the data-cube fast-Fourier transforms can be run along each dimension of the data-cube to generate range-measurements, range-rate measurements, and angle-of-arrival measurements from the frequency domain [11]–[13]. In most commercially available SoC radar, the resulting frequency graphs are post-processed by a constant false alarm rate filter [13], [14] which uses non-maximal suppression to determine sufficiently unique point targets published into a point cloud for end-use applications. As mentioned in Section I, these targets are subject to various sources of noise and are more sparse than other range-measurement sensors.

B. Related Works

Mapping: Approaches to radar-based mapping usually seek to address inherent radar challenges, such as susceptibility to false positives stemming from spurious returns and multipath reflections. [15] attempted to create an occupancy grid by learning an inverse sensor model, incorporating modeled variance to account for areas occluded in the sensor's field of view. This work also relied on 2D scanning radar. [4] used generative adversarial networks trained with lidar scans to generate radar grid-maps, which were robust to fog, and filled in areas from sparse radar measurements. [16] applied target presence probabilities to radar scans aligning them to map priors to generate 2D grid maps. [17] used a U-Net to learn super-resolution range-azimuth measurements for 2D grid-maps. Another recent method predicts 2D grid maps directly from radar range-azimuth heatmaps using diffusion models [9].

While the majority of these methods have primarily operated in 2D, more recent approaches have aimed to tackle the challenge of creating 3D maps. [8] learned depth representations from dense radar heatmaps, leveraging this information to generate 3D radar maps. [7] generated a full SLAM solution with correlated key points extracted from multiple scans used to generate 3D maps.

Our approach is focused on generating 3D maps using sparse radar points, rather than heatmaps, with common occupancy mapping methods such as Octomap [10]. This still creates two significant challenges: 1) the beam sensor model used in Octomap does not accurately model the operation of the radar sensor, and 2) radar false positives are still present. To address the first problem, we employ a new, radar-specific sensor model first presented in [3]. To address the second challenge, we leverage point cloud completion networks to upsample, fill, and denoise radar point clouds.

Pointcloud Completion Networks: Early point cloud networks such as PointNet, use multi-layer-perceptron layers to aggregate features through farthest point sampling (FPS) and achieve permutation invariance on unstructured point cloud [18], [19]. The first proposed point cloud completion network (PCN) uses convolutional encoders to transform the input point cloud to a k-dimensional feature vector, and the decoder utilizes this feature vector to generate both a coarse and detailed point cloud [20].

More recently transformers [21] have revolutionized many types of networks including point cloud completion networks. In their seminal method, PoinTr [22] extracts a coarse point cloud and features using DGCNN graph extractor and uses these resulting features in a geometry-aware transformer to compute set-to-set translational features for point cloud completion. These translational features are employed to generate coarse-to-fine point cloud upsampling steps using FoldingNet [23]. Improving on their method, Yu developed AdaPoinTr which uses the PoinTr backbone and introduces dynamic point query generation to rank the input point proxies and the encoder outputs, and denoised queries to make the decoder learn from high-quality queries [24]. To mitigate the memory constraints associated with analyzing full pointcloud, SeedFormer [25] extracts local coarse point cloud seeds and applies an upsampling transformer to gradually refine coarse-to-fine pointcloud through 3 stages. In UpPoinTr, we combine the base architecture of AdapoinTr with the point cloud region upsampling layers of SeedFormer as described in Section III.

III. Methods

Our method consists of three steps, as illustrated in Figure 2. First, a series of radar scans and a trajectory are passed through a probabilistic map generator. The resulting map is then divided into patches using FPS. Finally, these patches are passed through the AdaPoinTr, and re-assembled into a predicted lidar-like map. We provide further details of these steps in the following subsections. We also provide our code at `https://github.com/arpg/RMap`.

A. Probabilistic Voxel Map Generation

Since radar scans contain a limited number of points compared to standard 64-beam lidar scans which can contain up to 2^{16} points, directly upsampling radar scans to match lidar scans is challenging. Additionally, radar and lidar have different fields of view (FoV). Therefore, our first step is to create an intermediate representation bridging the gap

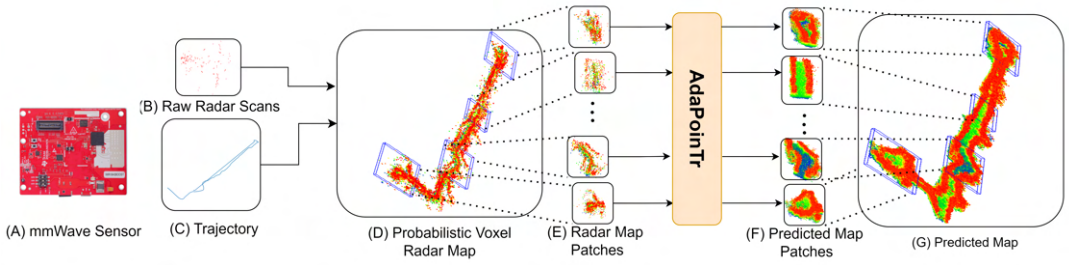

Fig. 2: System Diagram of RMap. An (A) automotive-grade system-on-chip mmWave radar sensor produces (B) Radar point clouds generated by CFAR thresholding along a trajectory (C). A (D) Radar Map is generated using the radar point clouds with Radar Octomap along the trajectory of the robot, with different pose locations highlighted. This map is then divided into (E) Patches of size 2048 sampled at different locations along the trajectory. These patches are passed through the pointcloud completion network, AdaPoinTr, to produce (F) predicted map patches. The predicted point clouds of size 8192 from AdaPoinTr are then merged into (G) the final predicted radar map. This method is described in Section III.

between radar and lidar. Maps serve as this intermediate representation allowing for the accumulation of various scans into the structures of their relative environments. Voxel-based maps allow us to generate a space-efficient representation of the world. We specifically use Octomap [10] for both lidar and radar-based maps.

To accurately compare these maps, we first apply a filter to the lidar, restricting the FoV to 180° to match the FoV of the radar. Subsequently, we generate a lidar octomap with the filtered point cloud and a ground-truth trajectory. We then generate a radar map using the radar sensor model presented in [3]. This sensor model replaces the ray-cast model from octomap with an explicit radar occupancy model. Voxels within the radar FoV where radar targets are detected have their probability increased, while voxels in the FoV without returns have their probability decreased. This model more accurately captures occupied space considering the noisy yet penetrative capabilities of radar sensors [3].

We have observed that these radar maps consistently have fewer voxels than lidar maps. Across the ColoRadar dataset, lidar maps typically contained approximately 2x–7x more voxels than the radar map applying only the radar sensor model. To address this disparity, we have designed an upsampling and filling method with the aim of bridging this gap and generating lidar-like maps from radar scans.

B. FPS based Patch Generation

While the size of each radar octomap is dependent on the length and shape of its trajectory, most maps contain more than 2^{16} voxels. If each voxel's center point is a single input to the generative transformer, memory limitations on modern graphics cards make inputting the entire map infeasible. To effectively train and run the network at inference, we aim to generate patches containing orders of magnitude fewer points. Randomly generating these patches may leave some areas uncovered. Notably, with random sampling, region-growth algorithms selecting the nearest n-points from a given pose sample could miss edges and corners of the map.

Instead, we generate t seed points along the entire radar map using FPS. From each seed point, we use geometric region growing to sample either the N^L or N^R nearest lidar and radar points. These resulting sub-patches are accumulated into the set of lidar patches $\mathscr{P}_k^L$ and radar patches $\mathscr{P}_k^R$. These sets of patches cover a sufficient portion of the map, with all points being captured in at least one patch. We detail the algorithm formally in Algorithm 1. As radar maps contain noisy points more deviated from its actual location in groundtruth, and FPS considering the farthest points could result in these outliers. To reduce the outliers, we added a preprocessing step to initially remove the points that are too deviated from the center of each small region in entire map. As the maps can have dynamic length, we considered number of seed of points to be dependent on both the number of points in the input map and number of points in the patch, with a as a variable to increase the number of patches around each seed points. This is detailed in Algorithm 1.

Algorithm 1 FPS based Patch Generation

Input: Lidar Map: $\mathscr{P}^L = \{p_i | p_i \in \mathscr{R}^3 \text{ for } i = 1,...,n^L\}$
Radar Map: $\mathscr{P}^R = \{p_i | p_i \in \mathscr{R}^3 \text{ for } i = 1,...,n^R\}$
Lidar patch size: N^L
Radar patch size: N^R
Output: Lidar Patches: $\mathscr{P}_k^L \subseteq \mathscr{P}^L$
Radar Patches: $\mathscr{P}_k^R \subseteq \mathscr{P}^R$ for $k = 1,...,t$

Initialize Lidar-patches $\mathscr{P}_k^L = \{\}$, Radar-patches $\mathscr{P}_k^R = \{\}$
Generate seed points p_i by sampling t farthest points in $\mathscr{P}^R$ where $t = an^R/N^R$
for i = 1:t **do**
  Find N^L closest points to p_i of $\mathscr{P}^L$
  Append result to $\mathscr{P}_k^L$
  Find N^R closest points to p_i of $\mathscr{P}^R$
  Append result to $\mathscr{P}_k^R$
end for
Return $\mathscr{P}_k^L$, $\mathscr{P}_k^R$

C. Dataset

For our experiments, we used the ColoRadar dataset [6] which contains a suite of radar and lidar point clouds on a

Environment	# Scenes	L/R Ratio	D-Dist [50%, 75%, 90%, 95%] (m)
arpg_lab	5	4.28	[0.150, 0.450, 0.808, 1.500]
aspen	12	2.59	[0.260, 0.450, 0.654, 0.862]
ec_hallways	5	6.68	[0.260, 0.367, 0.600, 0.765]
edgar_army	6	4.69	[0.260, 0.450, 0.636, 0.822]
edgar_classroom	6	5.27	[0.300, 0.450, 0.687, 0.925]
outdoors	9	3.87	[0.474, 0.875, 1.374, 1.736]
Overall	44	4.39	[0.335, 0.497, 0.862, 1.152]

TABLE I: Comparison of lidar and radar maps on the ColoRadar dataset. The ratio of points in the lidar map to the radar map L/R is provided along with the percentage of points contained within a certain deviation percentage in meters. For example, in the ARPG Lab Scene, 95% of radar points are within 1.5m of a lidar point.

diverse set of 44 scenes captured across 6 different indoor and outdoor environments. We generated voxel maps as described in Section III-A using a 0.15m resolution, a typical voxel size used in navigation applications [26], [27]. We then converted the maps into point clouds using the center of each occupied voxel in the map. We detail the number of scenes, along with the ratio of lidar map points to radar map points (L/R) for each environment in Table I.

Considering the overall L/R ratio was approximately 4, we generated input patches with 2048 points and ground-truth patches with 8192 points using pose sampling. For each environmnt labeled with run0-run(x), we consider run0-run($\lfloor \frac{x}{2} \rfloor$) for each environment as the test data and rest of the data for training. The test set would have 21 scenes for the total 44 scenes in the entire ColoRadar dataset, allowing to test unseen places for each environment. The train and test set ratio would be 52:48. Also, the patches generated using train scenes and test scenes have a ratio of 54:46, with the train and test scenes both containing different duration of the scenes.

D. UpPoinTr

We designed a custom upsampling point cloud completion network, UpPoinTr, which shares its core architecture with AdaPoinTr [24], utilizing the set-to-set translation strategy and the geometry-aware transformer architecture, and dynamic query generation blocks. We progressively enhance the coarse point cloud outputs $\mathscr{P}_{c,0}$ by passing each point cloud through a series of upsample layers inspired by the SeedFormer architecture [25]. Each upsampling layer generates a hierarchical series of point clouds, progressively increasing the resolution. Each Upsample layer takes the previous upsample layer's point cloud $\mathscr{P}_{c,i-1}$ and the predicted proxy features $\mathscr{F}_{i-1}$ from the previous layer to generate an upsampled point cloud $\mathscr{P}_{c,i}$ and $\mathscr{F}_i$. The first Upsample layer considers the $\mathscr{P}_{c,0}$ as the initial point cloud and the decoder features from AdaPoinTr $\mathscr{F}_0$ as the first feature set.

To account for the permutation invariance of the unstructured point clouds, we employ the Chamfer distance (CD) as the loss function [28], which has $O(N\log(N))$ complexity. To capture both the fine details and the overall structure at each upsampling layer, we consider CD loss for each intermediate point cloud generated. The ground truth for each of the intermediate point clouds (G_i) is generated by downsampling the ground truth point G to $|\mathscr{P}_{c,i}|$ using FPS. The resulting CD calculation appears as:

$$CD_i = \frac{1}{|\mathscr{P}_{c,i}|}\sum_{p\in\mathscr{P}_{c,i}}\min_{g\in G_i}\|p-g\|_2^2 + \frac{1}{|G_i|}\sum_{g\in G_i}\min_{p\in\mathscr{P}_{c,i}}\|p-g\|_2^2 \tag{1}$$

The overall loss J is:

$$J = \sum_i CD_i. \tag{2}$$

We used UpPoinTr with a set of hyperparameters similar to those used in the AdaPoinTr backbone. A notable exception is that AdaPoinTr added in 64 noise points to their queries induced in the Dynamic Query bank. We instead set the number of added points to 0. The resulting coarse point cloud generated from AdaPoinTr $\mathscr{P}_{c,0}$ has 512 points. As a result, we use the upsampling factors of [1, 4, 4] in the upsample layers to generate the final pointcloud with 8192 points.

We trained AdaPoinTr and UpPoinTr end-to-end with AdamW [29] optimizer with an initial learning rate of 10^{-5} and weight decay of 5×10^{-5}. The batch size is set to 16, and trained for 100 epochs with a continuous learning rate decay of 0.9 for every 20 epochs.

IV. Results

A. Evaluation Metrics

We consider the CD-l_1, CD-l_2, and F-Score, commonly used in evaluating the point completion networks on PCN and ShapeNet dataset. Lower CD metrics and higher F-Scores are considered better. We calculate CD-l_1 as CD metric described in [20] and CD-l_2 as described in [28] and equation 1, following the representations from [22], [25]. Both CD metrics are scaled by 1000, as is typical in other evaluations [20], [22], [24], [25]. We also calculated the F-Score following [30]:

$$P(d) = \frac{1}{|P|}\sum_{p\in P}[\min_{g\in G}\|p-g\|_2 < d] \tag{3}$$

$$R(d) = \frac{1}{|G_i|}\sum_{g\in G}[\min_{p\in P}\|p-g\|_2 < d] \tag{4}$$

$$\text{F-Score}(d) = \frac{2P(d)R(d)}{P(d)+R(d)} \tag{5}$$

where $P(d)$ denotes the Precision and $R(d)$ denotes the Recall of the predicted points within a distance of d from the ground truth points. In this equation, P denotes the final predicted point cloud and G denotes the ground truth point cloud. We considered $d = 0.15m$ to estimate the predictions falling in with-in the same voxel.

	Total Inference time(s)	**Patch CD-l_1 (m)**	**Patch CD-l_2 (m)**	**Patch F-Score**	**Map CD-l_1 (m)**	**Map CD-l_2 (m)**	**Map F-Score**	**Downsampled Map CD-l_1 (m)**	**Downsampled Map CD-l_2 (m)**	**Downsampled Map F-Score**
AdaPoinTr [24]	**183**	0.241	0.309	**0.548**	0.147	0.137	0.729	**0.214**	**0.279**	**0.590**
UpPoinTr (Ours)	423	**0.224**	**0.256**	0.546	**0.140**	**0.121**	**0.738**	0.221	0.293	0.574

TABLE II: Relative performance of AdaPoinTr compared with UpPoinTr on the ColoRadar dataset. Patch-level performance is the overall performance on the patches from the test set. Map-level performance is overall performance on test scene maps combining the points from those patches of each scene. Downsampled Map performance is the overall performance of all test scene maps that are downsampled to the map resolution of 0.15m. Total Inference time is calculated on NVIDIA RTX 6000 processing each of the patch sequentially.

Test Scene	**Input CD-l_1 (m)**	**Input CD-l_2 (m)**	**Input F-Score**	**Input D-Dist [50%, 75%, 90%, 95%] (m)**	**RMap CD-l_1 (m)**	**RMap CD-l_2 (m)**	**RMap F-Score**	**RMap D-Dist [50%, 75%, 90%, 95%] (m)**
arpg_lab_run0	0.329	0.651	0.297	[0.212, 0.367, 0.618, 0.808]	0.182	0.223	0.674	[0.072, 0.095, 0.184, 0.360]
arpg_lab_run1	0.347	0.639	0.247	[0.260, 0.424, 0.618, 0.750]	0.210	0.296	0.618	[0.074, 0.094, 0.143, 0.231]
aspen_run0	0.500	1.184	0.187	[0.335, 0.474, 0.687, 0.862]	0.218	0.269	0.575	[0.080, 0.112, 0.210, 0.321]
aspen_run1	0.500	1.212	0.217	[0.300, 0.450, 0.618, 0.735]	0.299	0.594	0.490	[0.078, 0.105, 0.177, 0.256]
aspen_run2	0.641	1.931	0.202	[0.300, 0.474, 0.779, 1.061]	0.231	0.415	0.577	[0.038, 0.070, 0.164, 0.268]
aspen_run3	0.540	1.514	0.241	[0.212, 0.367, 0.636, 0.900]	0.198	0.205	0.586	[0.076, 0.085, 0.111, 0.161]
aspen_run4	0.556	1.546	0.204	[0.300, 0.474, 0.687, 0.849]	0.178	0.207	0.671	[0.066, 0.082, 0.122, 0.171]
aspen_run5	0.511	1.315	0.210	[0.260, 0.450, 0.636, 0.875]	0.192	0.181	0.620	[0.084, 0.101, 0.172, 0.267]
ec_hallways_run0	0.426	0.717	0.143	[0.367, 0.618, 0.900, 1.142]	0.233	0.312	0.581	[0.098, 0.125, 0.237, 0.418]
ec_hallways_run1	0.493	0.992	0.127	[0.367, 0.618, 0.925, 1.219]	0.256	0.353	0.481	[0.082, 0.144, 0.276, 0.435]
edgar_army_run0	0.370	0.712	0.222	[0.300, 0.450, 0.618, 0.765]	0.201	0.297	0.627	[0.077, 0.095, 0.160, 0.255]
edgar_army_run1	0.374	0.720	0.245	[0.260, 0.450, 0.618, 0.779]	0.217	0.305	0.604	[0.070, 0.099, 0.195, 0.333]
edgar_army_run2	0.386	0.704	0.238	[0.260, 0.450, 0.687, 0.912]	0.174	0.163	0.621	[0.072, 0.088, 0.149, 0.248]
edgar_classroom_run0	0.388	0.786	0.229	[0.300, 0.474, 0.704, 0.925]	0.167	0.124	0.678	[0.093, 0.107, 0.159, 0.251]
edgar_classroom_run1	0.379	0.806	0.225	[0.300, 0.450, 0.687, 0.960]	0.176	0.132	0.641	[0.091, 0.120, 0.189, 0.291]
edgar_classroom_run2	0.332	0.609	0.279	[0.212, 0.367, 0.636, 0.887]	0.154	0.148	0.711	[0.062, 0.086, 0.150, 0.249]
outdoors_run0	0.540	1.136	0.130	[0.424, 0.765, 1.255, 1.650]	0.235	0.311	0.550	[0.091, 0.113, 0.216, 0.394]
outdoors_run1	0.640	1.527	0.097	[0.541, 1.006, 1.602, 2.012]	0.278	0.388	0.479	[0.093, 0.141, 0.346, 0.659]
outdoors_run2	0.573	1.194	0.108	[0.474, 0.875, 1.391, 1.781]	0.191	0.225	0.608	[0.059, 0.112, 0.203, 0.308]
outdoors_run3	0.635	1.396	0.093	[0.541, 0.972, 1.537, 1.909]	0.261	0.365	0.481	[0.076, 0.142, 0.298, 0.622]
outdoors_run4	0.555	1.075	0.0123	[0.450, 0.808, 1.209, 1.507]	0.239	0.344	0.509	[0.069, 0.122, 0.232, 0.366]
Overall	0.477	1.065	0.193	[0.335, 0.541, 0.875, 1.181]	0.214	0.279	0.590	[0.080, 0.107, 0.197, 0.330]

TABLE III: Comparison of input radar maps and the RMap or the maps generated from the outputs of AdaPoinTr w.r.t lidar maps. Overall RMaps outperform the input radar map, improving the overall Chamfer Distance metrics, F-Score (with a deviation threshold of 1 voxel i.e 0.15) and reducing the percentage of points in the radar map that occur outside 0.15m or one voxel of the nearest lidar point.

Metrics like CD-l_2, CD-l_1 only provide an average distance of the predicted points from the groundtruth, and are affected by the outliers [30]. We introduce a new metric to analyze the deviation distribution over the entire map, providing deeper insights on how the final predicted map points are scattered compared to the groundtruth lidar map.

$$D = \{\min_{p \in P} \|p - g\|_2\} \quad \forall g \in G \tag{6}$$

Deviation distribution (D-Dist) considers the distance x(m) where a percentage of ground-truth points are separated by at most that distance to the closest point in the predicted or input pointcloud. We present the distribution of this deviation D at 50, 75, 90, 95 percentiles. For instance at the 75th percentile if D-Dist is 0.5m, then 75% of groundtruth points have a predicted point match a maximum distance of 0.5m away. So, lower deviation distribution metrics are considered better.

As a baseline for comparison between our original dataset and our predicted radar maps, we present the deviation distribution(D-Dist) metric for the entire ColoRadar dataset in Table I. The metric provides the distribution of the distance from a point in the lidar point cloud to the nearest radar point in the radar point cloud, as defined in Equation 6. Based on this metric $< 50\%$ of the points in the initial radar point cloud are contained within the voxel resolution of 0.15m. At the 75th percentile most points in the radar map are at least 3 voxel widths apart at 0.45m.

To understand the deviation in different dimensions, we also present Chamfer Distance along each dimension, which is calculated as follows:

$$CD^x = \frac{1}{|P|} \sum_{p \in P} \|p^x - g^x_{min}\|_2^2 + \frac{1}{|G|} \sum_{g \in G} \|p^x_{min} - g^x\|_2^2 \tag{7}$$

where p_{min} represents the 3D point in predicted pointcloud P, such that it has the closest distance to given 3D point g in

Map	CD-l_2 (m)	CD^x (m)	CD^y (m)	CD^z (m)
Input Map	1.065	0.267	0.223	0.576
RMap	0.279	0.115	0.083	0.080

TABLE IV: Overall Comparison of chamfer distance along each dimension for the input radar map and the RMap w.r.t lidar groundtruth maps.

groundtruth pointcloud G and p^x represents the x-coordinate of 3D point p. Similarly, we calculate CD^y and CD^z.

B. Quantitative Results

We initially compared the performance of UpPoinTr with the AdaPoinTr, both at the patch-level and the map-level and presented the results in Table II. Although UpPoinTr marginally improves over the AdaPoinTr at the patch-level, the performance is almost similar at the map-level. After combining the patches, we downsample the pointcloud to the original map resolution (0.15m), which considers the average of all the points that are observed in a 0.15x0.15x0.15 grid. This results in slightly altering the closest points to the groundtruth, and affecting the overall performance. The marginal improvements with the UpPoinTr at the patch level are not observed after downsampling the pointcloud. Also, with the upsampling blocks added at the end of the AdaPoinTr backbone in UpPoinTr has further increased the inference time to 2x. As such, we consider the more efficient AdaPoinTr and overall downsampled map-level performance for our further experiments.

We present the metrics CD-l_1, CD-l_2, F-Score and Deviation Distribution (D-Dist) in Table III for the input test maps and the RMaps, the maps generated from combining the patches predicted from AdaPoinTr and then downsamled to the map resolution of 0.15m. Through these metrics, there is a clear increase in the number of points falling within the chosen voxel resolution of 0.15 with most environments maintaining this level of overlap out to the 75th percentile. Additionally, extreme errors, e.g. radar voxels over three voxel widths away from the nearest lidar voxel, occur around the 95th percentile in most maps. We analyzed the deviation along the each dimension using CD^x, CD^y, CD^z for the input map and the final RMap for all the test maps and presented them in Table IV. The performance along the low-resolution dimensions of mmWave radar i.e y, z greatly improved with RMap

We also analysed how a, the parameter in computing number of seed points t in each map, affects the overall performance in Table V. With only limited patches, the network is presented only with fewer patches to train on and not fully optimized, leading to lower performance. We observe that with $a = 16$, the performance of the network is much higher, but the total inference time to process all the patches would also significantly increase with the increased number of patches. So, we used $a_{train} = 16$ to generate patches for the maps in the train set and train the AdaPoinTr network. We tested this pretrained model on various test sets, generated by varying a_{test}, and results are presented in Table VI. With a_{test} =4, we observed better performance both in inference runtime and CD-l_1, $CD-l_2$, $F-Score$. The inference times for $a_{test} = 4$ are much lower, when compared with the total duration of all the test scenes combined (3400s), which makes it run real-time only single-GPU machines. The inference times can be further improved by processing the patches in parallel.

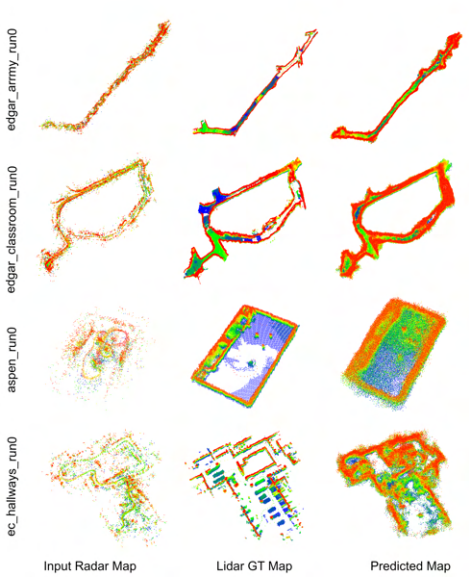

Fig. 3: Comparison of generated maps with naive radar accumulation (1st column), lidar (2nd column), and our technique (3rd column). For all maps we have removed the ceiling at an arbitrary 0.8m height.

C. Qualitative Results

We show several example maps generated from merged predicted patches in Figure 3. In general, the predicted maps successfully resemble the lidar maps – tracking walls, free space, and clutter accurately.

More specifically we note in Figure 4 the explicit benefits and some minor drawbacks of our method. This figure shows the cross-section of the original radar map, the lidar ground truth map, and the predicted radar map. Through this cross-section analysis, we see that the original radar map consists primarily of noise. However, after training it has a similar structure to the lidar map, distinguishing between free space and occupied space.

We also note an example degenerate case in our method in Figure 5. Our method acts as an infilling predictor. In this outdoor scene, the radar only detects points on the ground along the trajectory for a significant portion of the map. In the predicted map, these points along the floor are infilled with some additional geometry not present in the lidar groundtruth map. Conversely, the portion of trajectory in the structured area still performs well. We discuss some potential solutions to this in Section V.

a	CD-l_1 (m)	CD-l_2 (m)	F-Score	D-Dist [50%, 75%, 90%, 95%] (m)	Total patches
1	0.463	1.050	0.302	[0.249, 0.557, 1.271, 1.915]	669
2	0.304	0.457	0.438	[0.154, 0.261, 0.509, 0.835]	1363
4	0.239	0.302	0.545	[0.110, 0.176, 0.338, 0.553]	2749
8	0.222	0.273	0.581	[0.093, 0.130, 0.244, 0.406]	5515
16	0.214	0.279	0.590	[0.080, 0.107, 0.197, 0.330]	11062
32	0.225	0.323	0.563	[0.080, 0.100, 0.157, 0.256]	22127

TABLE V: Performance of RMap framework with varied number of patches based on a. In this comparison, we considered same a for both train and test sets.

a_{test}	CD-l_1 (m)	CD-l_2 (m)	F- Score	D-Dist [50%, 75%, 90%, 95%] (m)	Total Test patches	Total Inference Time (s)
1	0.349	0.822	0.487	[0.208, 0.623, 1.449, 2.082]	307	12
2	0.231	0.307	0.589	[0.128, 0.235, 0.507, 0.849]	624	28
4	0.209	0.239	0.617	[0.103, 0.161, 0.317, 0.508]	1262	56
8	0.207	0.248	0.611	[0.087, 0.128, 0.243, 0.397]	2532	92
16	0.214	0.279	0.590	[0.080, 0.107, 0.197, 0.330]	5075	183
32	0.227	0.326	0.557	[0.076, 0.097, 0.165, 0.281]	10160	370

TABLE VI: Performance of RMap framework on varied number of patches in test set based on a_{test}. Pretrained AdaPoinTr model on patches from training set with $a_{train} = 16$ is used for comparison. Inference time is calculated on NVIDIA RTX 6000 processing each of the patch sequentially.

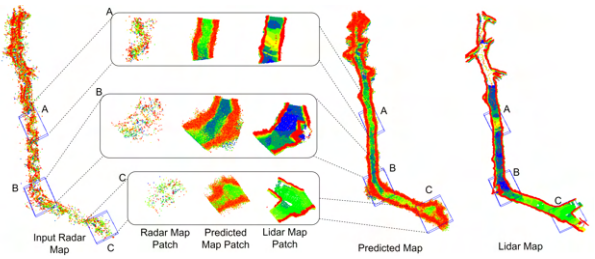

Fig. 4: Detailed view of various cross sections on a tunnel (edgar_army) map. (A) represents a cross-section of the maps along the straight path, (B) represents a cross-section of the maps at a corner, (C) represents a cross-section of the maps at a multi-path. The input radar map contains significant noise and missing regions and boundaries. Lidar map depicts the actual structure of the environment with minimal noise. The predicted Map using RMap mostly reconstructs the boundaries while significantly reducing noise. For all maps we have removed the ceiling at an arbitrary 0.8m height.

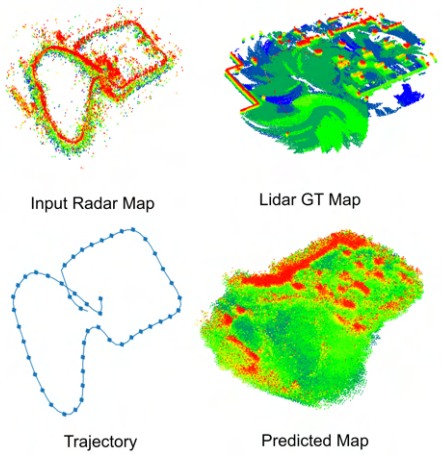

Fig. 5: An example degenerate case from RMap. As our method acts as an infilling method in some regions, along the trajectory, in otherwise free space, RMap generates geometry along the ground (green) instead of maintaining a flat region. This can be seen in the elevation of the predicted map colored from low (green) to high (red). This is likely a result of only having radar points generated along the ground. In the structured portions of the environment, RMap still sufficiently reconstructs the map. For all maps we have removed the ceiling at an arbitrary 0.8m height.

V. Discussion

We find RMap to be a compelling method for upsampling, filling, and denoising radar map points. The value of RMap becomes evident in the qualitative results, as depicted in Figure 3. In structured environments, RMap represents the nearby structure of various test environments. A more detailed examination of a tunnel map, as presented in Figure 4, reveals that while the original map is not suitable for navigation but our noisy predicted map has the potential for navigability. Clear distinctions between free and occupied space are apparent, which were absent in the original radar map.

While RMap performs well in structured areas, it can encounter challenges in regions with little structure. We note in Figure 5 that along empty sections of the trajectory, RMap generates geometry in the same shape as the original trajectory. In the more structured parts of the environment, however, the predicted radar map is still sufficiently structured. We acknowledge that some future work, pre-processing the radar maps prior to input into our network, or an alternate training strategy with more explicit representations for planar features such as large flat spaces may further enhance our method.

In order to achieve the highest patch-based accuracies Up-

PoinTr shows some marginal benefits. Future investigations are required to translate the patch-based accuracy into overall map-based accuracy and improve the efficiency of the added upsampling layers. While our RMap met many important metrics, it is possible that pre-filtering data to remove noise, clutter, or mutli-path reflections either through traditional processing or through data-driven approaches could provide better results. This is something we plan to explore through future work.

VI. Conclusion

We have presented a novel method, RMap for generating high-resolution 3D radar maps, which can run real-time on single GPU machines. As mentioned in Section V, the resulting RMaps produce radar maps that maintain similar structure to lidar maps, with the ability to operate in potentially visually degraded environments. RMap is demonstrated to capably separate free and occupied space as well as remove outliers using radar alone. Future work may further address separating empty regions and identifying key geometry either prior to input into the network or as a constraint within the network to avoid infilling improperly. We plan to include radar based odometry using [7], [31], rather than completely relying on lidar-based odometry for the initial trajectory and also improve the estimated odometry based on the RMap to provide a complete radar-based SLAM.

VII. Acknowledgements

This work was supported by USDA NIFA/NRI 2021-67021-33450.

References

[1] K. Ebadi, L. Bernreiter, H. Biggie, G. Catt, Y. Chang, A. Chatterjee, C. E. Denniston, S.-P. Deschênes, K. Harlow, S. Khattak, *et al.*, "Present and future of slam in extreme underground environments," *arXiv preprint arXiv:2208.01787*, 2022.

[2] K. Burnett, Y. Wu, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Are we ready for radar to replace lidar in all-weather mapping and localization?," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10328–10335, 2022.

[3] A. Kramer and C. Heckman, "Radar-inertial state estimation and obstacle detection for micro-aerial vehicles in dense fog," in *Experimental Robotics: The 17th International Symposium*, pp. 3–16, Springer, 2021.

[4] C. X. Lu, S. Rosa, P. Zhao, B. Wang, C. Chen, J. A. Stankovic, N. Trigoni, and A. Markham, "See through smoke: robust indoor mapping with low-cost mmwave radar," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, pp. 14–27, 2020.

[5] K. Harlow, H. Jang, T. D. Barfoot, A. Kim, and C. Heckman, "A new wave in robotics: Survey on recent mmwave radar applications in robotics," *arXiv preprint arXiv:2305.01135*, 2023.

[6] A. Kramer, K. Harlow, C. Williams, and C. Heckman, "Coloradar: The direct 3d millimeter wave radar dataset," *The International Journal of Robotics Research*, vol. 41, no. 4, pp. 351–360, 2022.

[7] Y. Zhuang, B. Wang, J. Huai, and M. Li, "4d iriom: 4d imaging radar inertial odometry and mapping," *IEEE Robotics and Automation Letters*, 2023.

[8] R. Xu, W. Dong, A. Sharma, and M. Kaess, "Learned depth estimation of 3d imaging radar for indoor mapping," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13260–13267, IEEE, 2022.

[9] R. Zhang, D. Xue, Y. Wang, R. Geng, and F. Gao, "Towards dense and accurate radar perception via efficient cross-modal diffusion model," *arXiv preprint arXiv:2403.08460*, 2024.

[10] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[11] C. Iovescu and S. Rao, "The fundamentals of millimeter wave sensors," *Texas Instruments*, pp. 1–8, 2017.

[12] M. Richards, J. Scheer, J. Scheer, and W. Holm, *Principles of Modern Radar: Basic Principles, Volume 1*. Institution of Engineering and Technology, 2010.

[13] M. Richards, W. Melvin, J. Scheer, and J. Scheer, *Principles of Modern Radar: Advanced Techniques, Volume 2*. Institution of Engineering and Technology, 2012.

[14] F. HM, "Adaptive detection mode with threshold control as a function of spatially sampled clutter-level estimates," *Rca Rev.*, vol. 29, pp. 414–465, 1968.

[15] R. Weston, S. Cen, P. Newman, and I. Posner, "Probably unknown: Deep inverse sensor modelling radar," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 5446–5452, IEEE, 2019.

[16] Y. S. Park, J. Kim, and A. Kim, "Radar localization and mapping for indoor disaster environments via multi-modal registration to prior lidar map," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1307–1314, IEEE, 2019.

[17] A. Prabhakara, T. Jin, A. Das, G. Bhatt, L. Kumari, E. Soltanaghai, J. Bilmes, S. Kumar, and A. Rowe, "High resolution point clouds from mmwave radar," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4135–4142, IEEE, 2023.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[20] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 international conference on 3D vision (3DV)*, pp. 728–737, IEEE, 2018.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "Pointr: Diverse point cloud completion with geometry-aware transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12498–12507, 2021.

[23] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 206–215, 2018.

[24] X. Yu, Y. Rao, Z. Wang, J. Lu, and J. Zhou, "Adapointr: Diverse point cloud completion with adaptive geometry-aware transformers," *arXiv preprint arXiv:2301.04545*, 2023.

[25] H. Zhou, Y. Cao, W. Chu, J. Zhu, T. Lu, Y. Tai, and C. Wang, "Seedformer: Patch seeds based point cloud completion with upsample transformer," in *European conference on computer vision*, pp. 416–432, Springer, 2022.

[26] M. T. Ohradzansky, E. R. Rush, D. G. Riley, A. B. Mills, S. Ahmad, S. McGuire, H. Biggie, K. Harlow, M. J. Miles, E. W. Frew, *et al.*, "Multi-agent autonomy: Advancements and challenges in subterranean exploration," *arXiv preprint arXiv:2110.04390*, 2021.

[27] H. Biggie, E. R. Rush, D. G. Riley, S. Ahmad, M. T. Ohradzansky, K. Harlow, M. J. Miles, D. Torres, S. McGuire, E. W. Frew, *et al.*, "Flexible supervised autonomy for exploration in subterranean environments," *arXiv preprint arXiv:2301.00771*, 2023.

[28] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.

[29] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2018.

[30] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3405–3414, 2019.

[31] H. Lim, K. Han, G. Shin, G. Kim, S. Hong, and H. Myung, "Orora: Outlier-robust radar odometry," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2046–2053, IEEE, 2023.