Towards Foundational Models for Single-Chip Radar

Supplementary Material

A. Data Collection and Dataset Details

Our data collection system, `red-rover`[4], is fully open source. The data collection rig and control app are shown in detail in Fig. 11, along with the system in its bike-mounted configuration in Fig. 10; all parts used can be purchased either off-the-shelf or 3D-printed using an ordinary 3D printer.

Bill of Materials The total cost of the bill of materials for our data collection system is $4,440 for the base system (Table 6). Note that the Lidar is the primary contributor to the cost of our data collection rig; while we use an OS0-128 ($12,000), it can be substituted for an OS0-64 ($8,000) or OS0-32 ($4,000) without any hardware or software modifications, though at the cost of reduced Lidar data quality.

A detailed bill of materials including all parts used in the data collection rig (along with CAD files for 3D printed parts) is available in our `red-rover` project repository.

Resource Usage For the configurations which we used to collect I/Q-1M, our data collection rig has the following overall characteristics:

- Data rate: ≈120GB/hour (≈33MB/s — 260Mbps), with some variation depending on the compressibility of the data. In practice, we do not find storage to be a substantial limitation, with the total dataset size being ≈3.5TB.
- Power consumption: ≈80W average. Using a 240Wh AC battery bank, this results in around 3 hours of battery life.

A.1. Sensors

Our data collection rig includes a radar, lidar, camera, and IMU, and records a total data bitrate of ≈260 Mbps. Almost half of the bitrate is consumed by the radar (126 MBbps), with the remainder being split between the Camera and Lidar, with a negligible amount data recorded from the IMU.

Radar "Boost" development boards for the TI 77GHz single-chip mmWave radar family[5] are commonly used in academic research, and we are not aware of any raw single-chip radar datasets – or tooling for data collection – which uses other radars. As such, we use the AWR1843Boost Radar (and a DCA1000EVM capture card), which is commonly used in prior literature [18, 29, 45].

The AWR1843Boost has 3 transmit (TX) and 4 receive (RX) antennas, resulting in 8 azimuth and 2 elevation bins (Fig. 12). We configured our radar to record 256 range

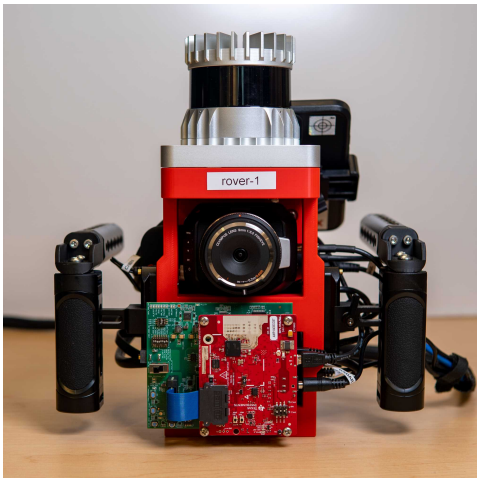
Figure 10. **Our data collection system, `red-rover`, in its bike-mounted configuration.** The sensors are mounted to a front rack, while the support electronics are mounted in the center frame and the battery at the rear for balance and stability.

Table 6. **Bill of materials and approximate cost of major components** as of time of writing, in US Dollars; carrying equipment (e.g., backpack, E-bike) and miscellaneous items under $100 (e.g. cables, screws) are not listed.

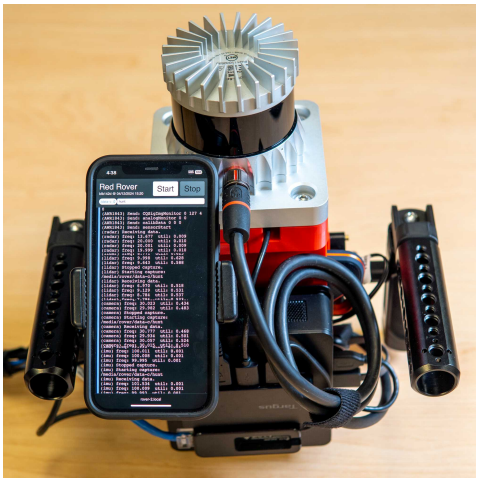
Item	Cost
Ouster OS0-32/64/128 Lidar	$4,000-$12,000
Data Collection Computer	$1,000
Black Magic Micro Studio Camera	$1,000
Magewell USB-SDI Capture Card	$300
OM Systems 9mm f/8 Fisheye Lens	$100
TI DCA1000EVM Radar Capture Card	$600
TI AWR1843Boost Radar	$300
XSens MTi-3 AHRS Development Kit	$450
External Storage Drive	$330
Battery	$240
Hardware for Handles	$120
Total	$4,440 + Lidar

[4] As the successor to our previous `rover` data collection system [18], `red-rover` is named for its distinctive red color.

[5] TI produces "Boost" series development boards across its range of 77GHz radars including the AWR1843Boost, the largely identical IWR1843Boost, and the AWR1642Boost, which is equivalent to the AWR1843Boost with its middle transmit antenna removed.



(a) Data collection rig from the front; the Lidar, Camera, and Radar (red PCB) along with its capture card (green PCB) are visible, while the IMU is hidden inside the red (3D-printed) plastic structure.



(b) Data collection rig from behind, showing the control app; the app allows users to specify metadata, then `start` and `stop` data collection. A live console displays logged messages and errors for each sensor.

Figure 11. **Close-up views of the handheld data collection rig from the front and back.**

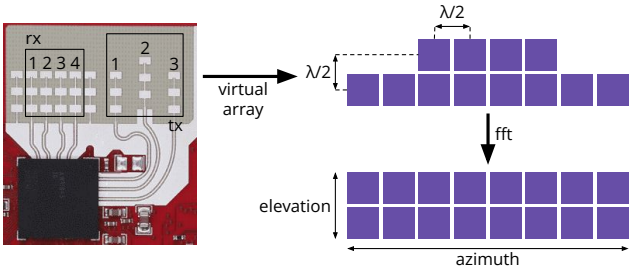


Figure 12. **Antenna configuration of the TI AWR1843Boost radar**. The 12 virtual antennas (top right) created by the radar's 3TX × 4RX antenna array (left) result in 2 elevation and 8 azimuth bins (lower right).

bins and 64 Doppler bins at 20 Hz, with varying range and Doppler resolutions depending on the setting; see Table 7 for detailed specifications. In our dataset, we also collect raw, uncompressed I/Q streams which are quantized as 16-bit integers by the radar; with the modulations used in our dataset, the radar has a total bitrate of 126 Mbps.

Crucially, we do not to use a high-resolution imaging radar (e.g., the 12 × 16 antenna TI MMWCAS-RF-EVM): in addition to their larger size, weight, and power, imaging radars have an order-of-magnitude higher raw data rate (e.g., ≈2gbps for an equivalent modulation using the TI MMWCAS-RF-EVM), which substantially increases the engineering and infrastructure cost of collecting raw data, while also making continuous live streaming and real-time deployment impractical.

Lidar We use an Ouster OS0-128 recording 2048 azimuth bins (1024 forward-facing) and 128 elevation beams at 10Hz. In practice, we find that the OS0 Lidar has a maximum range of 20-25m: while points further away can still *sometimes* be detected, objects are not consistently detected. As such, while our radar can detect objects at much further ranges, the Lidar forces us to restrict the maximum range used in our dataset; we plan to acquire a longer-range Lidar for future iterations of our dataset.

The lidar depth is LZMA-compressed, resulting in a bitrate of 14 Mbps. While we do not use these channels in our paper, we also collect the reflectance and near infrared background intensity, with a typical data rate of 9 Mbps and 22 Mbps, respectively.

Table 7. **Full radar configurations for each setting.** With a fixed frame size of $N_r = 256$ samples/chirp and $N_d = 64$ chirps/frame, configuring the radar's chirp rate, ADC sample rate, and chirp slope determines the range resolution $\Delta R = \frac{F_s c}{2N_r}$ and Doppler resolution $\Delta D = \frac{\lambda}{2N_d T_c}$ along with maximum range $R_{\max} = \frac{F_S c}{2s}$ and maximum Doppler $D_{\max} = \frac{\lambda}{4T_c}$, where λ is the radar's wavelength (77GHz; $\lambda = 3.9$mm) and c is the speed of light.

Setting	`indoor`	`outdoor`	`bike`
Chirp Time T_c	777μs	537μs	120μs
Sample Rate F_s	5MHz	5MHz	10MHz
Chirp Slope S	67MHz/μs	34MHz/μs	34MHz/μs
ΔR	4.4cm	8.7cm	8.7cm
$R_{\max}$	11.2m	22.4m	22.4m
ΔD	3.8cm/s	5.6cm/s	24.9cm/s
$D_{\max}$	1.2m/s	1.8m/s	8.0m/s

Camera We use a Black Magic Micro Studio Camera with an OM Systems 9mm f/8 Fisheye lens, recording at 1080p, 30 fps; frames are recorded as a MJPEG video, resulting in a typical bitrate of 88 Mbps. To minimize motion

Table 8. **Comparison with other mmWave radar datasets with raw data (4D data cube or equivalent), and a selection of other large datasets without raw data.** A *frame* in our table refers to the number of unique radar-sensor tuples. Our dataset is significantly larger than previous radar datasets, enabling us to scale up training.

Radar Type	Dataset	4D Data Cube	Dataset Size	Other Sensors
Single Chip	**IQ-1M (Ours)**	**Yes**	**29 hours (1M frames)**	Lidar, Camera, IMU
	MilliPoint [7]	No (3D Points)	6.3 Hours (545k frames)	Depth Camera
	nuScenes [6]	No (3D Points)	5.5 hours (400k frames)	Lidar, Camera, IMU, GPS
	RaDICal [29]	**Yes**	3.6 Hours (394k frames)	Depth Camera, IMU
	Coloradar [23]	**Yes**	2.4 hours (82k frames)	Lidar, IMU
	CRUW [63]	No (2D Map)	3 hours (400k frames)	Stereo Cameras
	RadarHD [45]	**Yes**	200k frames	Lidar
	CARRADA [41]	No (3D Cube)	21 Minutes (13k frames)	Camera
	RADDet [69]	**Yes**	10k frames	Camera
Cascaded	Radatron [33]	No (3D Cube)	4.2 hours (152k frames)	Camera
	K-radar [42]	**Yes**	35k frames	Lidar, Camera, IMU, GPS
	RADIal [49]	**Yes**	20k frames	Lidar, Camera, GPS
Mechanical	Oxford Radar RobotCar [3]	No (2D Image)	17 Hours (240k Frames)	Lidar, Camera, GPS
	RADIATE [54]	No (2D Image)	5.0 hours (72k frames)	Lidar, Camera, GPS
	Boreas [5]	No (2D Image)	350km	Lidar, Camera

blur, the camera is set to 18 db gain; the shutter speed is set to automatic. Note that while our camera and capture card are capable of 60 fps recording, we record only 30 fps since we find that 30 fps recording is far more stable than 60 fps (especially with regard to dropped frames), and since since the downstream tasks which we envision cannot easily take advantage of 60 fps video.

IMU We include a XSens MTi-3 IMU which is used for Cartographer SLAM in conjunction with our Lidar. The IMU records acceleration, angular velocity, and rotation at 100 Hz, with a total bitrate of 35 Kbps.

A.2. Time Synchronization

While each sensor is recorded against the same system clock, we asynchronously record each at its full "native" speed. To generate radar-lidar-camera samples, we align higher frequency sensors (radar – 20Hz; camera – 30Hz) to the Lidar (10Hz) by selecting the nearest sample in time to each lidar frame. Since our data collection implementations for each sensor have variable initialization and de-initialization time, we also trim regions at the start and end of each trace which do not have coverage from all sensors.

A.3. Comparison with Other Datasets

Table 8 enumerates a number of radar datasets sorted by radar type, along with their sizes and included sensors. Since different types of radars have substantially different operating modes, modulations, and data characteristics and dimensions, we focus on single-chip radars. In this category, prior datasets generally use TI single-chip radars such as the TI AWR1843 family which we use [59].

Fine-Tuning Experiment In our fine-tuning experiments, we elected to use the Coloradar [23] dataset due to its inclusion of high-quality Lidar depth data and extensive prior work using this dataset. We considered, but opted not to use, the following datasets:

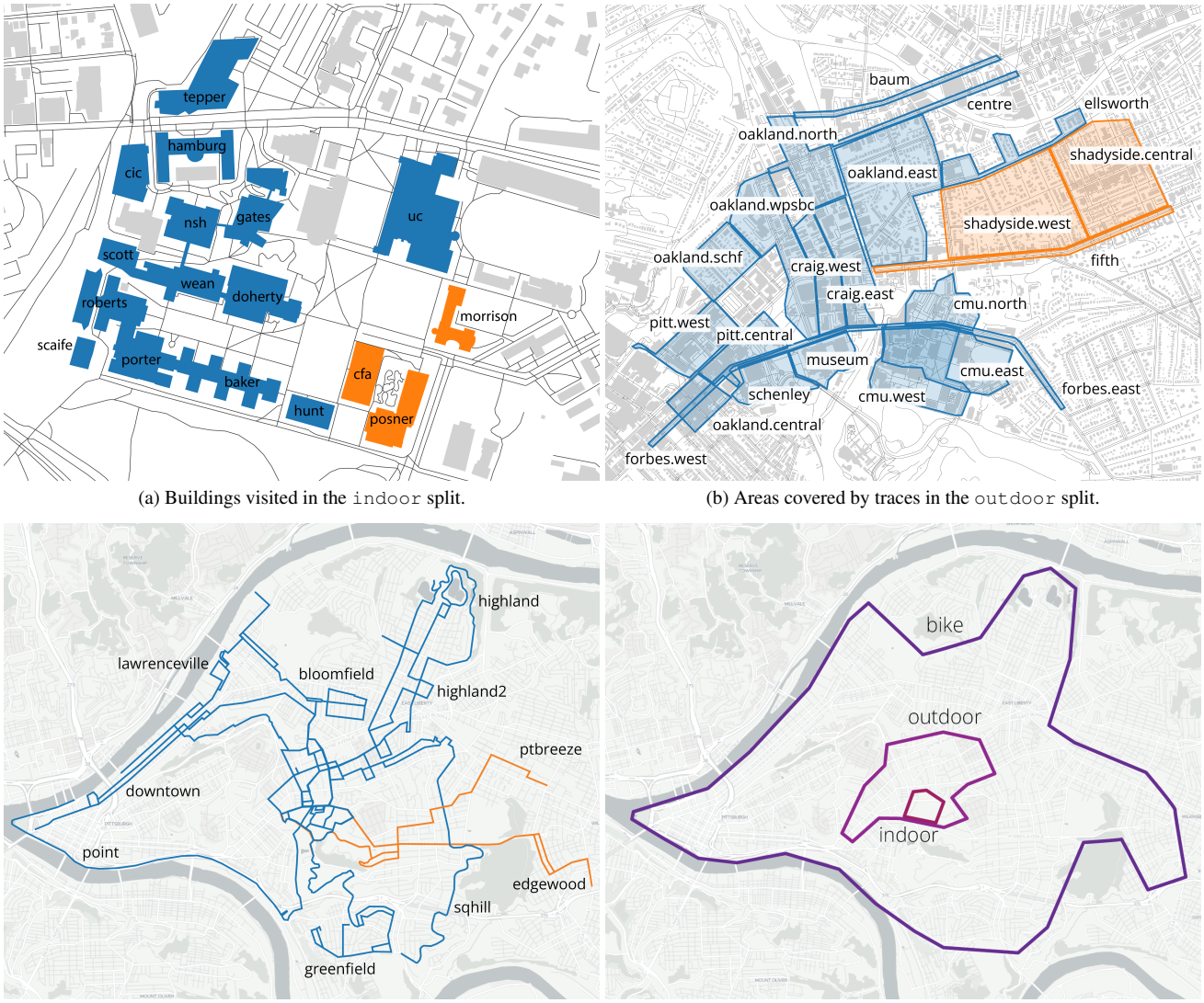
- RaDICal [29]: the depth cameras used have poor performance, especially beyond very close ranges. Likely because of this issue, we are also not aware of a substantial body of prior work using this dataset as a benchmark.
- RADDet [69]: RADDet is an extremely popular object detection dataset, and a good candidate for fine-tuning. Unfortunately, while we have been able to obtain the raw ADC data, the ground truth labels, video, and other metadata are no longer available as of time of writing.
- CRUW [63]: While the original CRUW dataset appears to include lower-level data, only 2D range-azimuth maps are publicly available.

A.4. Dataset Details

Our dataset was collected on and around the CMU campus and in the Pittsburgh area, and includes three data settings: handheld indoors, handheld outdoors, and on a bike. In addition to maps of data collection areas (Fig. 13), we also provide representative samples from each dataset (Fig. 14).

`indoor`: The indoor setting was collected from publicly accessible areas within the CMU campus. Each trace generally represents a different floor (or multiple floors, in cases where each floor is relatively small). Additionally, each floor was covered twice: once in a forward-facing configuration (with the velocity mostly aligned with the radar's orientation), and once in a "sideways facing" configuration (with the velocity mostly orthogonal to the radar).

`outdoor`: Roughly 30-minute-long traces were collected within contiguous areas with minimal overlap, with the radar generally facing forward. The areas visited include CMU and Pitt university campuses, commercial areas rang-



(a) Buildings visited in the `indoor` split.

(b) Areas covered by traces in the `outdoor` split.

(c) Traces in the `bike` split.

(d) Overview of the data collection extents for each setting.

Figure 13. **Maps of the train (blue) and test (orange) splits for each setting.**

ing from medium to high density, and residential areas ranging from single-family detached to high rise apartment buildings, as well as a variety of streets ranging from small alleys to busy "stroads."

`bike`: Data was collected on approximately 60-minute-long round-trips[6] originating from our lab; each trace is split into an inbound and outbound leg covering mostly the same path, but in different directions. Note that there is some overlap between the areas covered in the train and test splits at "bottlenecks" near the CMU campus; when viewed as a whole, we believe this is negligible.

[6]As one the authors was struck by a vehicle while collecting data on bike, we urge any efforts to replicate or extend this split to minimize mental load during data collection and use caution when planning routes. Thankfully, the author was uninjured, though the radar was destroyed.

B. Method Details

GRT uses a standard encoder-decoder transformer network (App. B.1). We also document our data augmentations (App. B.2), training tasks (App. B.3), evaluation procedure (App. B.4), and baselines (App. B.5).

B.1. GRT Training & Hyperparameters

GRT uses a standard transformer architecture with sinusoidal positional encodings, and experimentally obtained radar-specific patching parameters. For a summary of key hyperparameters and architecture parameters, see Table 9.

Architecture Unless specified otherwise, GRT's architecture uses the following common parameters:

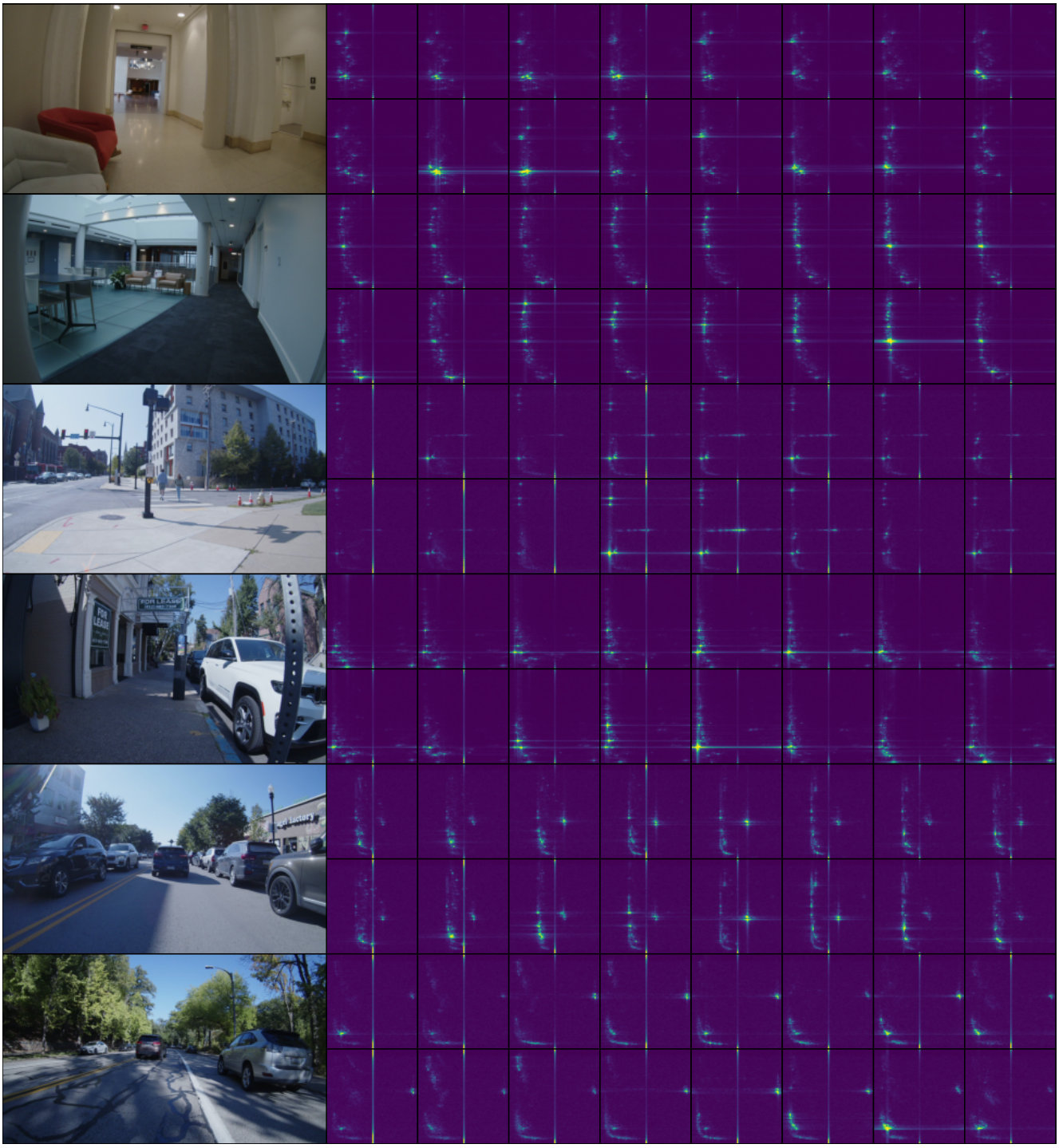


Figure 14. **Representative samples from our dataset showing camera and range-Doppler frames from the `indoor` (top), `outdoor` (middle), and `bike` (bottom) settings**. Each radar plot shows the range-Doppler image of a single (azimuth, elevation) bin. When the radar is configured with a range and Doppler resolution which is appropriate for each setting, the resulting range-Doppler frames are remarkably similar at a visual level. Note that common types of radar noise and artifacts such as a zero doppler artifact (the straight line at the center of each frame) and range-Doppler bleed (horizontal and vertical lines coming from bright reflectors) are clearly visible in these examples.

Table 9. **Key Hyperparameters for GRT.** Except for model layers and dimensionality, which we perform scaling law ablations on, these hyperparameters are taken from common transformer design practices as of time of writing.

Input Patch Size	128 $(4 \times 2 \times 8 \times 2)$
Input Number of Patches	2048
Output Number of Patches	1024
Layers	4 – 18
Model Dimension	256 – 768
Dimensions Per Head	64
Expansion Ratio	4.0
Transformer Norm	"pre-norm"
Activation	GeLU
Dropout	0.1
Batch Size	32
Optimizer	AdamW
Warmup	100 Steps
Learning Rate	10^{-4}

- We always use a simple linear patch and unpatch layers, with the appropriate output dimensionality depending on the task.
- All layers use a GeLU activation [16].
- Transformers use "pre-norm" (norm before, instead of after the transformer layer), which is generally regarded as more stable [65]; when using "post-norm" (as in the original transformer architecture [61]), we find that GRT often diverges due to numerical instability at initialization.
- Each transformer layer has a fixed expansion ratio of 4.0 and a dropout of 0.1.

Positional Encodings In both encoder and decoder positional embeddings, we use a simple N-dimensional encoding which divides the number of features equally between each dimension and independently applies a sinusoidal encoding for the coordinate in that axis. To facilitate fine-tuning for tasks with different output resolutions, we also normalize the frequencies by the total length of each axis so that different resolutions result in the same frequency range.

Training When training GRT, we use the following parameters for all models:
- We apply a range of data augmentations, which we find to provide a $\approx 5\%$ benefit (App. B.2).
- We always use a fixed batch size of 32. When training on platforms with different GPU counts, the batch is split equally between each GPU.
- All models are trained with a learning rate of 10^{-4} using the AdamW [31] optimizer with a warmup period of 100 steps. We find this warmup period to be essential in order to avoid initialization instability and NaN gradients.
- Each model was trained until the validation loss stopped decreasing, as defined by three consecutive checkpoints without a decrease in validation loss, with two checkpoints taken each epoch.

Fine-Tuning Fine-tuning uses the same procedure as for training, including termination after the validation loss stops decreasing. The model is not frozen, with all weights being trainable during the tuning process. In cases where the output dimension does not match the input dimension (e.g., 8-channel one-hot classification outputs for the Semantic Segmentation objective vs. 1-channel binary classification outputs for occupancy objectives), the output layer is also re-initialized.

Patch Size We use a patch size of 128 bins (4 range, 2 Doppler, 8 azimuth, 2 elevation) in the encoder, resulting in 2048 input patches, and square (or cubic) patches for each output sized to maintain a fixed decoder sequence length of 1024 patches. Note that this "patches out" the azimuth and elevation axes in the encoder; while we empirically determined that this leads to the best performance on our dataset (Sec. 5.1), we expect the optimal patch dimensions to vary depending on the input radar resolution.

Patch Size Alternatives In addition to the above patch size, we tested the following alternatives:
- **Range-Doppler-azimuth-elevation**: carefully selecting our patch size to keep all four axes, we create 16 range $\times$ 8 Doppler $\times$ 8 azimuth $\times$ 2 elevation patches. This results in a $4.18 \pm 1.09\%$ increase in test loss.
- **Range-azimuth-elevation**: we eliminate the Doppler axis for 128 Range $\times$ 8 azimuth $\times$ 2 elevation patches. This results in a $6.27 \pm 1.10\%$ increase in test loss.
- **Doppler-azimuth-elevation**: we eliminate the Range axis (as much as possible) to obtain 64 Doppler $\times$ 2 range $\times$ 8 azimuth $\times$ 2 elevation patches. This results in a $6.22 \pm 1.11\%$ increase in test loss.

B.2. Data Augmentations

We develop a range of data augmentations, which we ablate in two groups: *Scale, Phase, and Flip Only*, and *Full* augmentations, which we use by default.

Scale, Phase, and Flip Only This group includes "simple" augmentations which can be calculated pixel-wise:
- `radar_scale`: random scaling applied to the magnitude of the 4D radar data cube, with log-normal distribution $\exp(\mathcal{N}(0, 0.2^2))$ clipped to $[\exp(-2), \exp(2)]$.
- `radar_phase`: random phase offset of $\text{Unif}(-\pi, \pi)$ applied to the phase of the 4D radar data cube (except in ablations where no phase information is provided to the model).

- `azimuth_flip`: random flipping (with probability 0.5) along the azimuth axis, i.e., swapping left and right. Note that this augmentation also affects the ground truth for each task.
- `doppler_flip`: random flipping (with probability 0.5) along the Doppler axis, i.e. swapping positive and negative Doppler. This is equivalent to reversing the direction of travel of the sensor and all other objects in the scene; as such, this augmentation also affects the ground truth velocity by reversing the velocity vector.

Note that we do not apply an `elevation_flip` since the ground is always down!

Full In addition to the above augmentations, we include augmentations which are equivalent to random cropping:

- `range_scale`: ranges are multiplied by a $\text{Unif}(1.0, 2.0)$ scale, with the radar data cube being cropped and scaled appropriately using a bilinear interpolation. The ground truth occupancy and Bird's Eye View are also scaled accordingly.
- `speed_scale`: Doppler velocities are multiplied by a log-normal $\exp(\mathcal{N}(0, 0.2^2))$ distribution, clipped to $[\exp(-2), \exp(2)]$. All scaling is done with bilinear interpolation. If the velocity is scaled down, we zero-fill any extra bins; if velocity is scaled up, we "wrap" the Doppler velocity around to emulate the ambiguity of Doppler velocity modulo the Doppler bandwidth. Finally, the ground truth ego-motion velocity is scaled to match.

B.3. Task Details

3D Occupancy Classification Our 3D polar occupancy task uses a binary cross-entropy loss on polar grid cells which are created by the product set of the radar's range resolution with the Lidar's azimuth and elevation resolution. The loss is further scaled and balanced for the following:

- To facilitate joint training between different radar modulation, we normalize distances by the radar range resolution, resulting in a fixed output grid for each setting.
- To correct for the sparsity of 3D occupancy grids, occupied cells are weighted greater (64.0) than unoccupied cells (1.0).
- Since polar occupancy cells are larger when further away, we correct for the cell size, which is proportional to r^2.

Finally, to manage the memory required by dense 3D prediction, we apply a $4\times$ range, $8\times$ azimuth, and $2\times$ elevation decimation, resulting in (64 range $\times$ 128 azimuth $\times$ 64 elevation) bins, which we output with 1024 $(8\times8\times8)$ patches. For decimated range-azimuth-elevation grid r, θ, ϕ and 0-1 occupancy Y^*, this corresponds to the following loss:

$$\mathcal{L}(\hat{Y}_{r,\phi,\theta}, Y^*_{r,\phi,\theta}) = r^2(1.0 + 63.0Y^*_{r,\phi,\theta})\text{BCE}(\hat{Y}_{r,\phi,\theta}, Y^*_{r,\phi,\theta}) \quad (1)$$

In addition to the test loss, we compute the Chamfer distance (by treating each occupied cell as a point), and the mean absolute error of depth estimates obtained by finding the first occupied cell along the range axis of our range-azimuth-elevation occupancy.

Bird's Eye View (BEV) Occupancy We use the same binary cross-entropy and Dice loss mixture as [45], and output a 256×1024 range-azimuth polar occupancy grid which corresponds to the native range resolution of the radar and the native azimuth of the Lidar, restricted to forward-facing bins. We output 1024 (16×16) patches.

In addition to the test loss, we also compute the Chamfer distance, using the same procedure as for 3D occupancy.

Semantic Segmentation We use the 640×640 output of `segformer-b5` [64], trained on ADE20k [71], as the ground truth for this task. We aggregate the ADE20k classes into eight broad categories (arranged by index):

0. `ceiling`: any structure viewed from below; mostly seen `indoors`.
1. `flat`: flat, walkable surfaces such as sidewalks and roads. Grass and other vegetation are excluded, and included in `nature` instead.
2. `nature`: vegetation and other natural items such as grass, shrubs, trees, and water.
3. `object`: miscellaneous small objects such as furniture which are not included in the `structure` category.
4. `person`: any person who is not inside a vehicle or riding a vehicle.
5. `sky`: the sky.
6. `structure`: large man-made items such as buildings, fences, and shelters.
7. `vehicle`: cars, trucks, buses, and other vehicles.

We output 1024 $(5\times5)\times8$ patches and train using a simple binary cross-entropy loss. Finally, in addition to the test loss, we also calculate the mIOU (mean intersection over union), accuracy, and top-2 accuracy.

Ego-Motion Estimation We fuse our IMU and Lidar data Cartographer SLAM [17], which we project back to the sensor's frame of reference to obtain Ego-Motion ground truth, and manually exclude regions where SLAM failed (typically due to a lack of Lidar features, e.g. bridges or tunnels, totaling $\approx 1\%$ of our dataset).

During training, we first normalize the velocity with respect to the Doppler resolution (i.e. measuring each component in Doppler bins); then, we use the l_2 loss

$$\mathcal{L}_{\text{Ego-Motion}} = ||\hat{v} - v^*||_2 \approx \sqrt{(\hat{v} - v^*)^T(\hat{v} - v^*) + \varepsilon} \quad (2)$$

where $\varepsilon = 1.0$ Doppler bins (with a typical magnitude of $16 \leq ||v^*||_2 \leq 32$) is included for numerical stability.

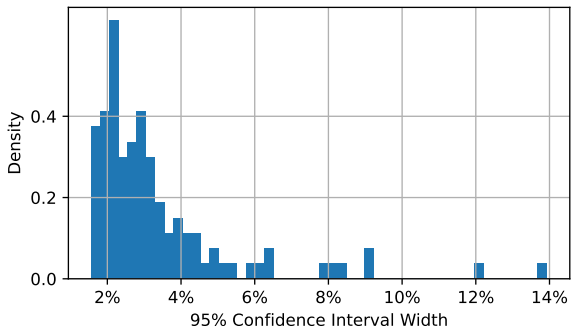


Figure 15. Width of 95% confidence intervals, in percent, relative to the baseline of each ablation, aggregated and plotted as a histogram. Using our 4.5 hour (163k frame) test split, we are able to compare methods with a median confidence interval width of 2.6% (one-sided difference of 1.3%), with the exact width varying depending on the variance of the underlying comparison.

B.4. Evaluation Procedure

Following our evaluation procedure, we can measure differences of 1-2% with high probability (Fig. 15); we provide details about this procedure below.

Geo-Split Within each setting, ≈1.5 hours of data are reserved as a test set. In order to control data leakage, we split traces for each setting along natural geographic boundaries:

- `indoor`: since buildings can have duplicated floor plates and other design features between floors or different areas, data was split by building, with the evaluation set consisting of all traces collected from 3 buildings.
- `outdoor`: each trace was collected as a contiguous area on foot; we reserved a set area within a neighborhood that includes various zoning and street types for the test set.
- `bike`: each trace was collected as a round-trip ride from a set origin; two rides from a set range of directions were reserved for the test set.

Sample Size Correction Intuitively, sampling the same signal – such as radar-lidar-video frames – with a greater frequency yields diminishing "information". Since the standard error of the mean, $\text{SE} = \text{std}(X)/\sqrt{N}$, is defined for N independent and identically distributed samples, we must correct for the *effective sample size* of our test data.

In our analysis, we assuming that changes in model performance imply changes in the underlying data (but not necessarily the converse). This allows us to estimate a lower bound on the effective sample size from each scalar performance metric as [51]

$$N_{\text{eff}} = \frac{N}{1 + 2\sum_{t=1}^{\infty} \rho_t} \tag{3}$$

for autocorrelation ρ_t (where t is the delay). Similar to [18], we approximate the infinite series up to $t = N/2$ and clip negative empirical autocorrelation values $\hat{\rho}_t < 0$ to 0.

Paired z-Test The actual values of each measured metric have a large inherent variance due to the varying difficulty of the data (e.g., the presence of clutter, dynamics, or other challenging features). As such, the standard error of *absolute* values of each metric is large. Taking advantage of the fact that each model is evaluated on identical test traces with respect to a baseline, and that the performance of each model is highly correlated with its baseline, we instead use a paired z-test, i.e., on the *relative* values of each metric, which mitigates the impact of this variance.

B.5. Baseline Details

Ideally, we would like to compare GRT against off-the-shelf baselines without any modifications. However, due to the lack of standardization in radar hardware and modulations, radar data cubes do not have standard dimensions and aspect ratios; furthermore, since radar data cubes are generally tightly coupled with physical effects arising from hardware and modulation design choices, dimension-normalizing transforms such as cropping and resizing are also not generally valid.

While transformer encoder-decoder architectures can be readily adapted for different input and output dimensions by simply changing the input and output context size (and positional encodings), convolutional and encoder-only architectures must be modified to fit different input and output resolutions. Thus, to run prior baselines on our dataset, we made the a few changes to each baseline.

RadarHD RadarHD [45] uses an asymmetric U-net with azimuth-only 2× upsampling layers to meet the target output resolution, relative to the input. Since RadarHD was originally designed for 512 output azimuth bins instead of the 1024 output azimuth bins in our dataset, we include an additional azimuth upsampling layer, with other layers and dimensions staying the same.

T-FFTRadNet T-FFTRadNet [15] uses a swin transformer encoder with a relatively lightweight convolutional decoder, with some U-net-like skip connections. Since T-FFTRadNet's "dense" high-resolution decoder was designed only for cascaded imaging radars, and the decoder for single-chip radar was designed only for sparse outputs, we modified the dense decoder to use the output of the backbone for single-chip radar by increasing the bilinear upsampling size to 4× in both range and azimuth. Other layers and dimensions are kept the same.

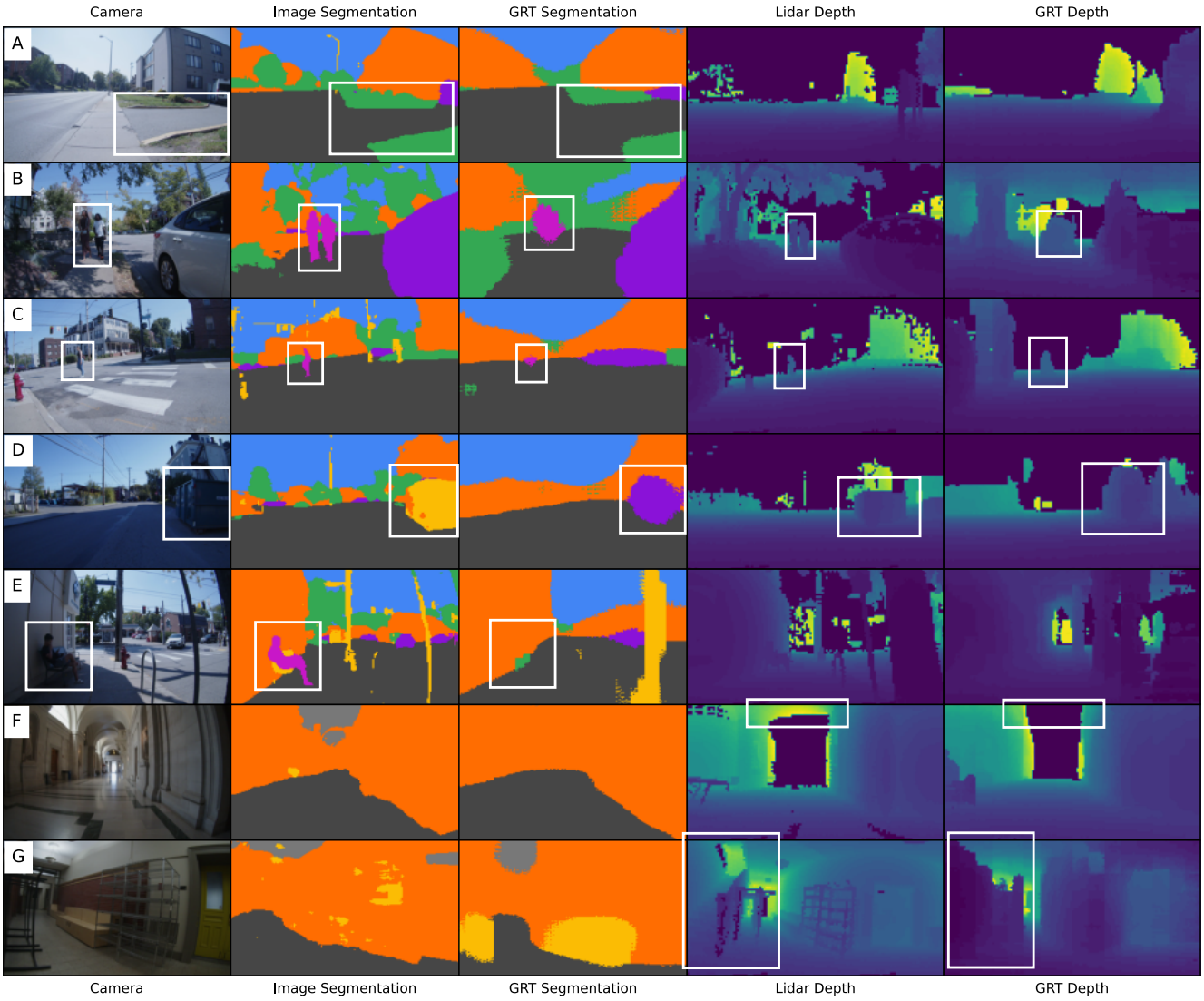


Figure 16. **Select frames from our test set**. Frames are annotated with notable features. Note that the field of view is narrower for the camera ($\approx 60^\circ \times 120^\circ$) compared to the Lidar ($90^\circ \times 180^\circ$).

C. Additional Results

In this section, we provide sample visualizations (App. C.1) and additional analyses (App. C.2-C.4). Note that in addition to the included figures, video examples of our model in action can be found at our project site.

C.1. Sample Images

To better visualize the capabilities of our model, we provide a range of sample results (Fig. 16), including some cases where our model performs better than expected, failure cases which illustrate the limitations of our approach, and a representative random sample (Fig. 17).

Surprising Capabilities While others have tried mmWave-radar-based semantic segmentation [24], no prior works attempt to extract high-resolution elevation information for tasks such as semantic segmentation or 3D occupancy classification on such a low-resolution radar. As such, we found it surprising that our model works at all! In our evaluation traces, we also found additional capabilities which further exceeded our expectations:

- **Material Properties**: Despite our radar's low resolution, it is able to correctly label pavement and grass in many cases (Fig. 16, A), likely learning the fact that paved surfaces generally have specular returns, while natural surfaces have diffuse returns.
- **Pedestrians**: the model is able to correctly identify pedestrians in many cases, likely due to the unique micro-doppler signature of people walking (Fig. 16, B-C).

Finally, it is worth noting that a radar using GRT's segmen-

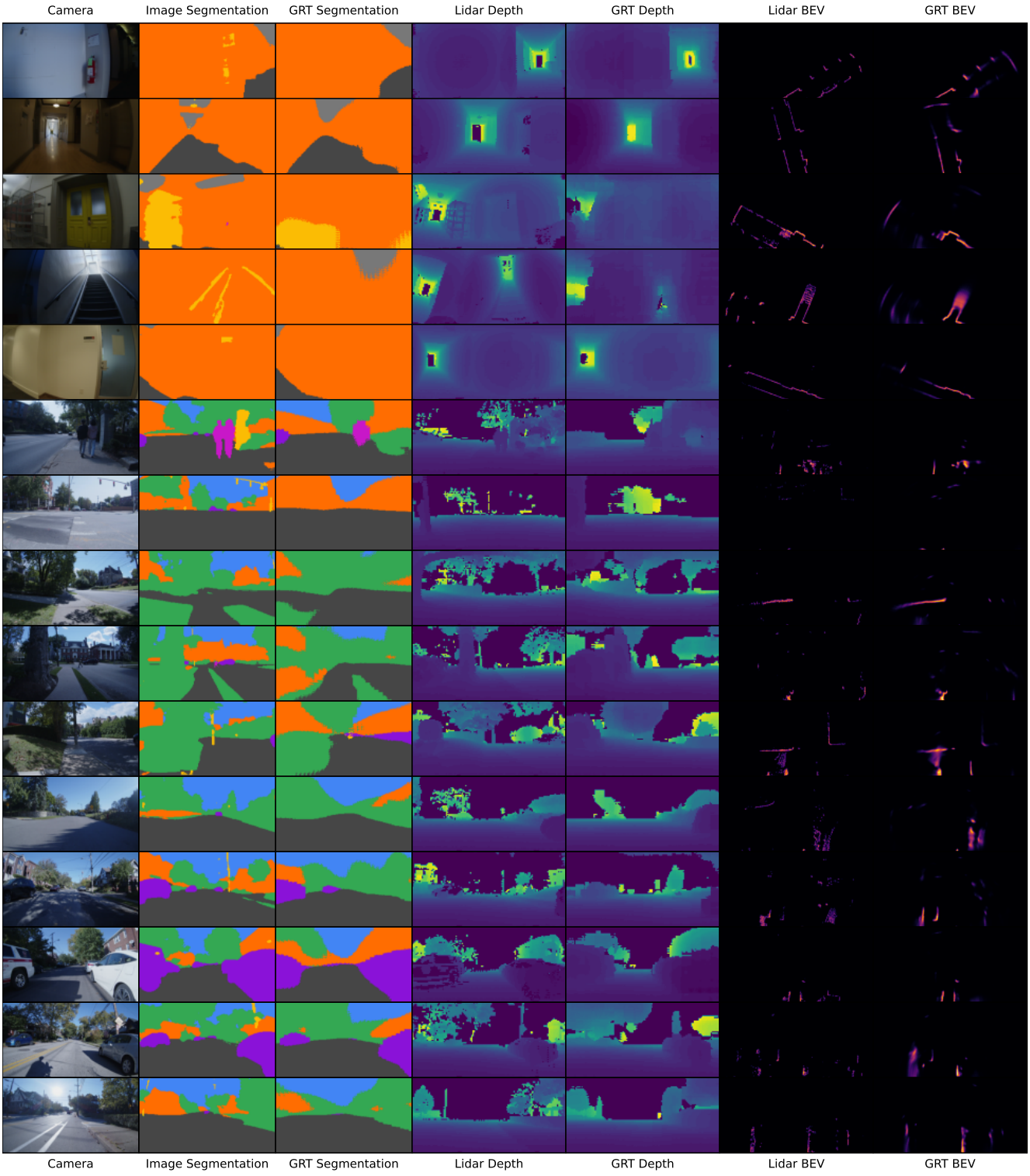


Figure 17. **A random sample of 15 images from the test split of I/Q-1M.** 5 samples are taken from the `indoor` (top), `outdoor` (middle), and `bike` (bottom) settings. All images are generated by `GRT-small`; the segmentation (left) and BEV (right) outputs are trained and fine-tuned on the full dataset, while the depth output (center) is rendered from the base model's 3D occupancy output.

Table 10. Performance metrics (App. B.3) of the `GRT-small` transformer model trained on our base task (3D Occupancy) and fine-tuned on each secondary task with our full dataset (mean with 95% confidence intervals).

Task	Metric	Average	Indoor	Outdoor	Bike
3D Occupancy	Chamfer	4.7 bins $\pm$ 0.19	0.24 m $\pm$ 0.02	0.4 m $\pm$ 0.019	0.38 m $\pm$ 0.023
	Depth	16 bins $\pm$ 0.66	0.62 m $\pm$ 0.059	1.5 m $\pm$ 0.09	1.4 m $\pm$ 0.089
Semantic Segmentation	mIOU	0.69 $\pm$ 0.012	0.78 $\pm$ 0.02	0.63 $\pm$ 0.019	0.69 $\pm$ 0.018
	Accuracy	0.79 $\pm$ 0.0097	0.85 $\pm$ 0.016	0.75 $\pm$ 0.016	0.78 $\pm$ 0.016
	Top-2 Accuracy	0.94 $\pm$ 0.0053	0.97 $\pm$ 0.006	0.92 $\pm$ 0.01	0.93 $\pm$ 0.011
BEV Classification	Chamfer	11 bins $\pm$ 0.91	0.28 m $\pm$ 0.027	0.84 m $\pm$ 0.13	1.3 m $\pm$ 0.19
Ego-Motion Estimation	Speed	0.95 bins $\pm$ 0.093	0.025 m/s $\pm$ 0.0022	0.025 m/s $\pm$ 0.0024	0.091 m/s $\pm$ 0.047
	Angle	4.2° $\pm$ 0.46	5.9° $\pm$ 0.54	5° $\pm$ 0.61	1.9° $\pm$ 1.3

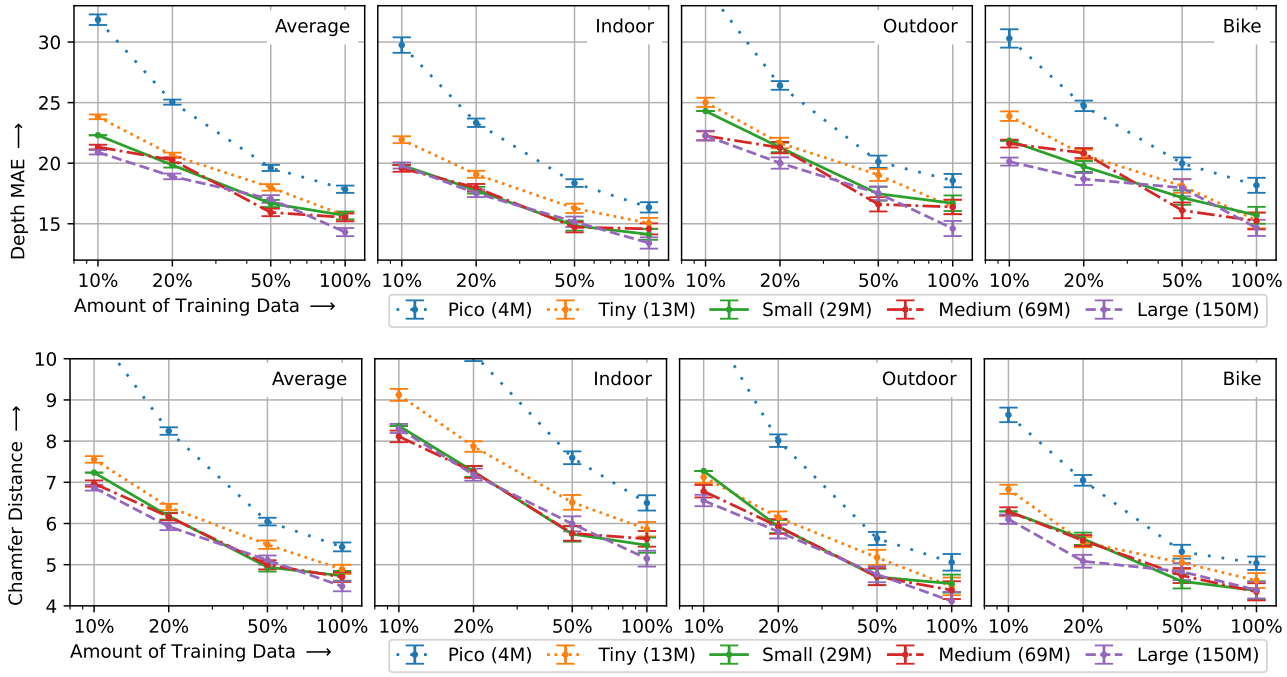


Figure 18. Scaling laws for the *depth* mean absolute error (top) and *chamfer* distance (bottom) metrics, measured in radar range bins (4.4cm `indoor`, 8.7cm `outdoor`, `bike`).

tation capability can operate in conditions when cameras cannot such as fog, smoke, and darkness.

Failure Cases To highlight a few failure cases for GRT:

- **Fine-Grained Classification**: using only a low-resolution radar without any visual or Lidar inputs, a pure radar transformer has no way of differentiating fine-grained classes such as metal dumpsters (in the `object` class) from the `vehicle` class (Fig. 16, D).
- **Static People**: without the unique micro-doppler signature associated with walking, our model often fails to detect people who are standing or sitting still (Fig. 16, E).
- **Limited Vertical Resolution**: the vertical field of view of our radar is relatively limited, with a 6dB-beamwidth of $\pm 20°$. Thus, even with Doppler information to help resolve elevation information (beyond the 2 elevation bins measured by our radar), the model cannot reliably estimate regions at the edge of the Lidar or Camera's vertical field of view (Fig. 16, F).
- **Clutter**: when a scene is very cluttered, GRT can fail to resolve individual objects; this generally leads to large hallucinations (Fig. 16, B, G).

C.2. Absolute Metrics

Since each task has a number of possible metrics (which are not always aligned), we generally report metrics as relative test losses to best capture the performance *of the model* in its ability to fit the target loss.

Key Metrics As an absolute reference, we calculated a range of common performance metrics for each objective

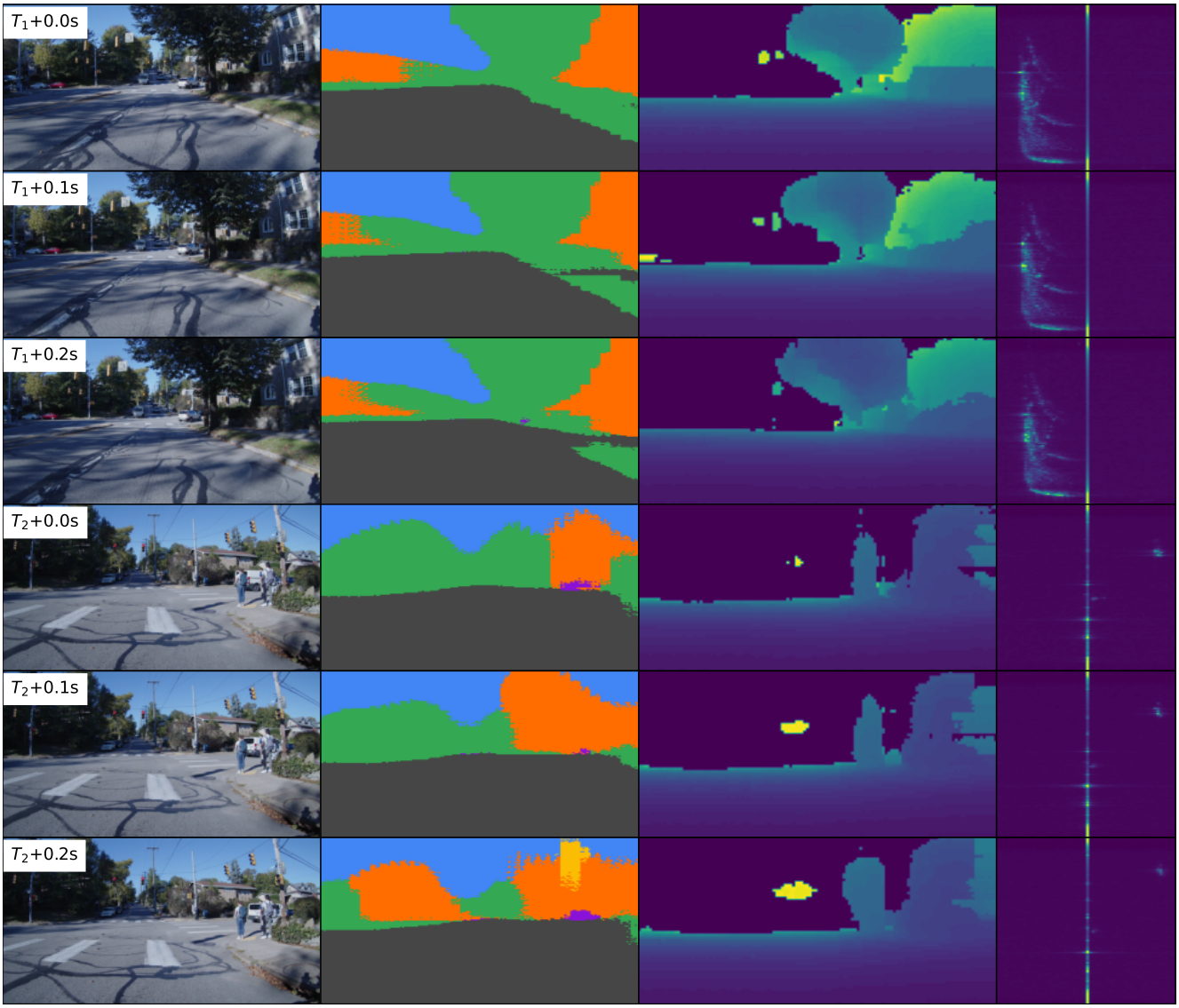


Figure 19. **Sample sequences of three consecutive frames before (top) and after (bottom) stopping at a red light.** Video frames are provided for reference (left), with GRT's semantic segmentation (center left) and depth (center right) outputs across the two sequences along with the range-Doppler spectrum. After stopping, the Doppler spectrum (horizontal axis; right) collapses to a single Doppler bin, resulting in significantly decreased information available to the model. This manifests as noisier (as seen by larger frame-to-frame variations and hallucinations) and less accurate predictions by the model.

(Table 10) as described previously (App. B.3). For these metrics, SI units are reported where applicable; distance and speed metrics are normalized by range and Doppler resolutions, respectively, when aggregating over settings with different radar configurations.

Absolute Scaling Laws As an alternate version to Fig. 6, we also measured scaling laws with respect to the *depth* and *chamfer* metrics (Fig. 18); while somewhat noisier, the same general trend can be seen. Note that this noise is also why we compare loss metrics in our scaling laws and ablations since it serves as a more direct measure of relative "learning" performance.

C.3. Impact of Doppler

To further illustrate the impact of Doppler, we measured the test loss of our objectives (other than Ego-Motion estimation), binned against the sensor speed (Fig. 20). When the sensor's speed is low relative to its maximum Doppler, it captures less Doppler information due to our radar's fixed Doppler resolution, leading to degraded performance. This can also be seen qualitatively: when the data capture rig

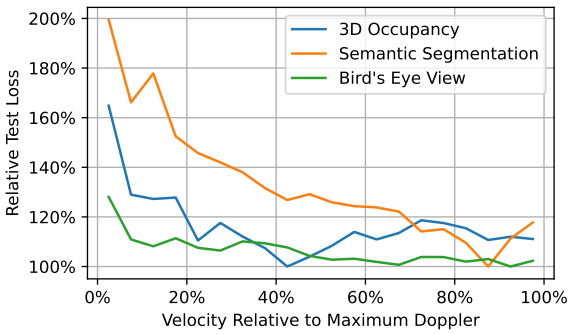


Figure 20. Relative test losses binned by the speed of the data collection rig (20 equal 5% bins) on the `bike` subset; due to less available Doppler information, lower speeds are associated with higher losses for each of our tasks.

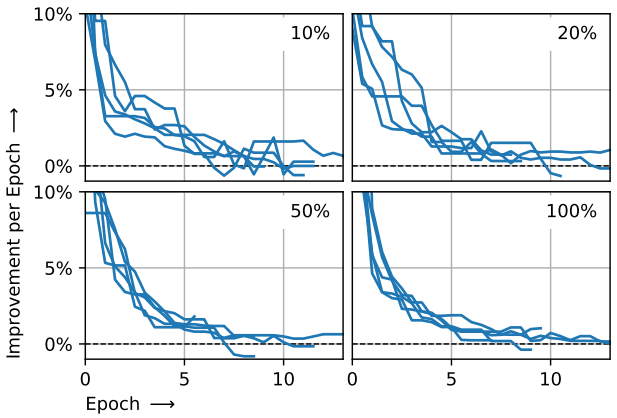


Figure 21. Validation loss improvement per epoch, measured every checkpoint (2 checkpoints/epoch), and smoothed with a 5-checkpoint median filter. Each line refers to a different model (with different sizes); models are separated by dataset size. Our models consistently tend to stop improving (in validation loss) after around 10 epochs of training.

stops, the GRT model's predictions become noisier, less sharp, and tend to show blocky artifacts aligned with the output patch size (Fig. 19).

C.4. Scaling Law Projections

To motivate future work scaling data collection and training for single-chip radar models using 4D data cubes, we run a suite of scaling law experiments to obtain rough, order-of-magnitude estimates for the data requirements of "fully" training GRT foundational model (Sec. 5.4) of a similar size to the relatively small (by vision standards) models which we trained. In this section, we describe additional details and assumptions behind our two methods of estimation.

Extending the Scaling Law In order to estimate data requirements from our scaling law, we start from the assumption that data scaling will always be at most logarithmic. This is based on the intuition that increasing the size of the dataset will always have diminishing returns (in turn at a diminishing rate), which is consistently observed in other scaling experiments [68].

We then train a network only on the test set, without any augmentations; we reason that this provides a lower bound on the achievable test loss (due to random, unpredictable noise in the dataset) for a given architecture, given that the model is not large enough to memorize the test set pixel-for-pixel. Note that this also provides an improvement over a naive lower bound from the fact that $\mathcal{L} > 0$.

Combining these two implies that data scaling can be logarithmic for increasing dataset size up to at most $100\times$ our current dataset size, which we believe represents a reasonable order-of-magnitude estimate for the data requirements for a "fully trained" radar foundational model.

Training Curve Patterns In our experiments, we observe that all models tend to stop improving with respect to validation loss after approximately 10 epochs (Fig. 21). This gives a further avenue for projections: assuming that the informational "value" of a radar frame is roughly equivalent to an image, we can take rough numbers for the typical number of samples seen used to train a vision transformer and translate this to data requirements for a radar foundational model.

Note that this assumption of informational equivalence is also quite rough. Unlike vision transformers, which are typically trained on *independent* images scraped from the internet, GRT is trained using *dependent* frames sampled from a time-series of sensor data, decreasing the relative information density of radar time-series data. On the other hand, while vision transformers typically use sparse feedback signals such as image-caption [47] or image-label [68] pairs, GRT is trained using dense feedback in the form of a 3D occupancy grid (App. B.3). In principle, this increases the relative information density of radar-Lidar training pairs. Our projection therefore assumes that these factors roughly balance out within an order of magnitude.