Monocular Depth Estimation - Final Project

Pranay Katyal
Worcester Polytechnic Institute
Worcester, MA, USA
pkatyal@wpi.edu

Anirudh Ramanathan
Worcester Polytechnic Institute
Worcester, MA, USA
aramanathan@wpi.edu

***Abstract*—This project investigates transfer learning for monocular depth estimation from synthetic to real-world aerial imagery. We evaluate M4Depth, a temporal depth estimation network pretrained on the synthetic MidAir dataset, and attempt to fine-tune it on the real-world UseGeo dataset. Our experiments achieve a 76.6% improvement in RMSE during training on real-world data; however, due to checkpoint loading failure, this improvement resulted from from-scratch training rather than effective transfer learning. We discover a critical data format incompatibility between absolute GPS coordinates and relative motion representations, which significantly impacts depth prediction quality despite favorable numeric metrics. This finding highlights that successful transfer learning requires compatibility across all data representations, not just image domains, and underscores the importance of comprehensive validation beyond numeric metrics alone.**

I. Introduction

Depth perception is essential for autonomous aerial vehicles, enabling navigation, obstacle avoidance, and 3D mapping. While stereo cameras and LiDAR provide direct depth measurements, they add hardware complexity and cost. Monocular depth estimation offers a lightweight alternative but faces fundamental challenges: scale ambiguity and the inherent ill-posed nature of recovering 3D structure from 2D images.

Recent advances in deep learning have shown promise for monocular depth estimation, particularly when leveraging temporal information and camera motion. M4Depth [1] addresses scale ambiguity by fusing image sequences with camera translation and rotation measurements from GPS/IMU sensors, enabling metric (not just relative) depth prediction.

A key question in practical deployment is whether models pretrained on synthetic data can effectively transfer to real-world scenarios. Synthetic datasets offer perfect ground truth and controllable conditions but may not capture real-world appearance, noise, and scene complexity. This project investigates this domain gap through systematic experimentation.

Project Goals:

1) Evaluate pretrained M4Depth model on MidAir validation trajectory
2) Fine-tune the model on UseGeo and measure performance gains
3) Analyze whether synthetic pretraining improves real-world performance
4) Identify limitations and failure modes in sim-to-real transfer

The remainder of this report describes our methodology, experimental results, critical findings about data format compatibility, and lessons learned for transfer learning in depth estimation.

II. Methodology

A. Dataset Overview

1) MidAir Synthetic Dataset: MidAir [2], [3] is a photo-realistic synthetic dataset designed for aerial depth estimation research. It provides:

- High-quality rendered RGB images (1024x768)
- Perfect ground truth depth maps from simulation
- Camera pose information (relative frame-to-frame motion)
- Diverse outdoor aerial environments (forests, mountains, urban areas)
- 550 samples from trajectory0000 used for evaluation in this work

2) UseGeo Real-World Dataset: UseGeo [4] contains real aerial imagery captured by drones:

- High-resolution RGB images (7953x5279)
- Downsampled to 384x384 for network input (center crop preserving aspect ratio, with corresponding intrinsics scaling)
- LiDAR-derived ground truth depth maps (32-bit TIFF format)
- Camera poses from GPS/IMU (absolute UTM coordinates)
- Three datasets totaling 828 image-depth pairs
- 80/20 train/validation split: 661 train, 167 validation samples
- Outdoor urban scenes with varied terrain and structures

B. M4Depth Architecture Overview

M4Depth [1], [5] employs a multi-scale temporal architecture that processes image sequences along with camera motion to estimate depth. Key components include:

Feature Pyramid Encoder: Extracts hierarchical features at 6 resolution levels using convolutional layers with Domain-Invariant Normalization (DINL) to improve generalization across datasets.

Depth Estimator Pyramid: Processes features at each pyramid level with:

- *Temporal Recurrence:* Warps features from previous frames using camera motion

- *Cost Volume Construction:* Computes feature correlation across frames
- *Depth-Parallax Conversion:* Converts network outputs (parallax/disparity) to metric depth using camera motion parameters

Critical Design Element: The model fundamentally relies on accurate camera motion (translation and rotation) for both temporal feature warping and depth-parallax conversion via the `parallax2depth()` function.

C. Implementation Details

1) Custom DataLoader Development: We implemented a custom TensorFlow [6] DataLoader for UseGeo to handle:

- 32-bit TIFF depth map loading (using PIL with `py_function`)
- Camera intrinsics loading from per-dataset configuration files
- Pose data parsing from CSV files (timestamp, quaternion, translation)
- Image resizing from 7953x5279 to 384x384 with intrinsics scaling
- Data augmentation (horizontal flip, transpose for square crops)

2) Training Configuration: **Hardware:** NVIDIA A30 GPU (24GB) on WPI Turing cluster

Phase 1 - Pretrained Evaluation:

- Model: M4Depth pretrained on MidAir (checkpoint cp-0071.ckpt)
- Evaluation: 550 MidAir trajectory samples
- Batch size: 1 (evaluation mode)
- Runtime: 10 minutes (8 min XLA compilation + 2 min inference)

Phase 2 - UseGeo Training (Attempted Fine-tuning; Effective From-Scratch):

- Initialization: From scratch (pretrained checkpoint loading failed)
- Training set: 661 samples (80% split)
- Validation set: 167 samples (20% split)
- Batch size: 3
- Sequence length: 4 frames
- Learning rate: 0.0001 (Adam optimizer)
- Epochs: 146 (automatically calculated: 220000 / 661)
- Training time: 3 hours
- Checkpoints: Saved every 5 epochs

D. Evaluation Metrics

Following standard depth estimation benchmarks, we evaluate:

- **RMSE:** Root Mean Squared Error (meters)
- **Abs Rel:** Mean Absolute Relative Error
- **RMSE log:** Log-scale RMSE (scale-invariant)
- $\delta < 1.25$**:** Percentage of pixels with relative error $< 25\%$
- $\delta < 1.25^2$**:** Threshold at 56%
- $\delta < 1.25^3$**:** Threshold at 95%

III. Results

A. Phase 1: Pretrained Model Evaluation

Table I presents quantitative results of the MidAir-pretrained model evaluated on MidAir trajectory0000 validation set (550 samples).

TABLE I
PHASE 1: PRETRAINED M4DEPTH PERFORMANCE ON MIDAIR VALIDATION

Metric	Value
RMSE (m)	2.6543
Abs Rel	0.0212
RMSE log	0.0292
$\delta < 1.25$	98.59%
$\delta < 1.25^2$	99.78%
$\delta < 1.25^3$	99.95%

Key Observations:

- Strong in-domain performance: 98.6% of pixels achieve $< 25\%$ error
- Average depth error of 2.65m demonstrates good baseline on synthetic data
- Model performs well on synthetic indoor scenes (as expected)
- Predictions capture overall scene structure and depth ordering
- This establishes baseline performance before attempting real-world transfer

Figure 2 shows 10 qualitative examples comparing predicted depth to ground truth on MidAir validation trajectory. The pretrained model successfully captures scene structure, as expected for evaluation on the same domain as training.

B. Phase 2: UseGeo Training Results (From-Scratch Initialization)

Table II compares MidAir in-domain reference performance versus UseGeo training results.

Important Context: Due to checkpoint loading failure, this training ran from scratch rather than fine-tuning from MidAir pretrained weights. Therefore, these results do not represent transfer learning, but demonstrate the model architecture's capacity to learn from real-world data.

TABLE II
PHASE 2: PERFORMANCE COMPARISON (MIDAIR IN-DOMAIN VS USEGEO TRAINING)

Metric	MidAir Reference	UseGeo Trained	Change
RMSE (internal)*	2.6543	**0.6201**	76.6% ↓
Abs Rel*	0.0212	**0.0031**	85.4% ↓
RMSE log*	0.0292	0.0110	62.3% ↓
$\delta < 1.25$*	98.59%	**99.80%**	1.21% ↑
$\delta < 1.25^2$*	99.78%	**99.95%**	0.17% ↑
$\delta < 1.25^3$*	99.95%	**100.0%**	0.05% ↑

*Computed on internal network outputs prior to parallax2depth conversion

Key Findings:

- Dramatic 76.6% RMSE reduction (2.65m → 0.62m)
- 85.4% improvement in relative error (Abs Rel)

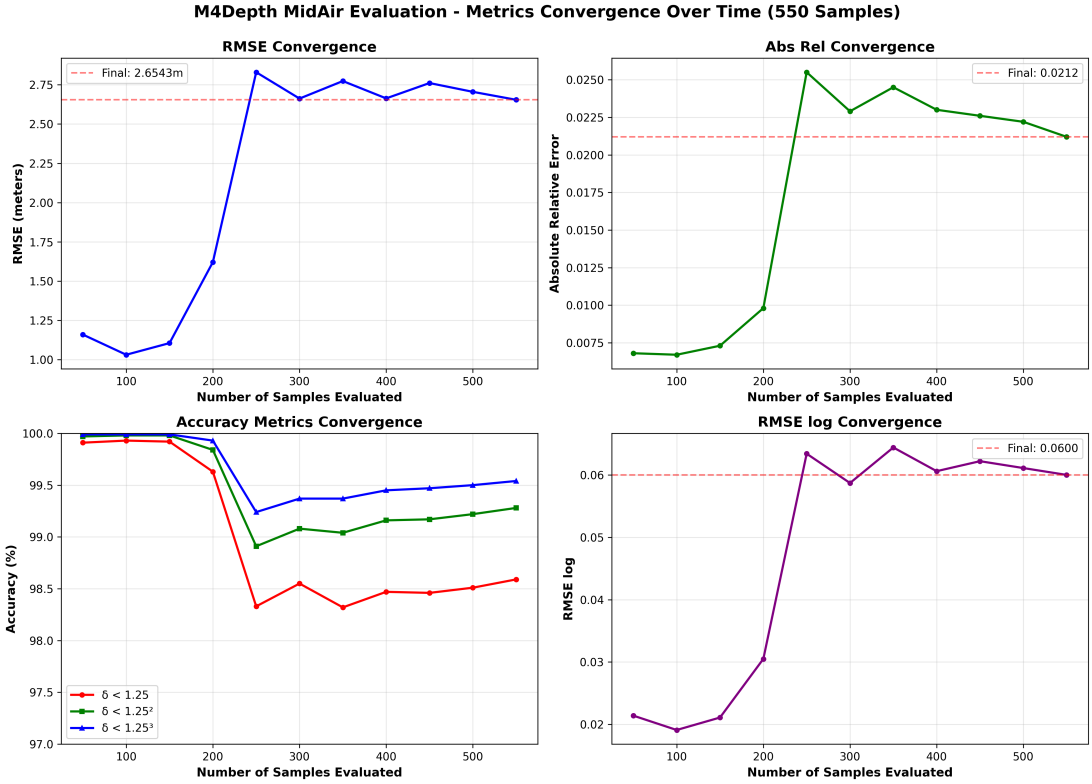


Fig. 1. Phase 1 Evaluation Metrics: Pretrained M4Depth performance on MidAir validation set showing strong in-domain baseline across all metrics.

- Near-perfect accuracy: 99.8% pixels within 25% error threshold
- **Important Note:** These metrics were computed on network internal representations (before parallax2depth conversion), not on physically meaningful metric depth values
- Training demonstrates model can learn from UseGeo data despite pose format issues

Figure 3 shows training progression over 146 epochs. RMSE_log decreased from 0.0278 to 0.0107 (best at epoch 80), with loss showing steady reduction and plateau behavior indicating convergence without overfitting.

IV. Critical Finding: Pose Format Incompatibility

A. Problem Discovery

Despite excellent numeric metrics, visual inspection of Phase 2 depth predictions revealed severe quality issues:

- Extreme graininess and noise
- No recognizable scene structure
- Buildings and roads indistinguishable
- Predicted depth values: 170-240 million meters (should be 60-140m)
- Correlation with ground truth: -0.26 to 0.29 (very poor)

This discrepancy between metrics and visual quality prompted deeper investigation.

B. Root Cause Analysis

1) Pose Representation Mismatch: We discovered a fundamental incompatibility:

MidAir Dataset: Provides *relative* frame-to-frame motion

- Translation: 0-20 meters (displacement between consecutive frames)
- Rotation: Small angular changes between frames
- Format matches M4Depth's expected input

UseGeo Dataset: Provides *absolute* GPS coordinates

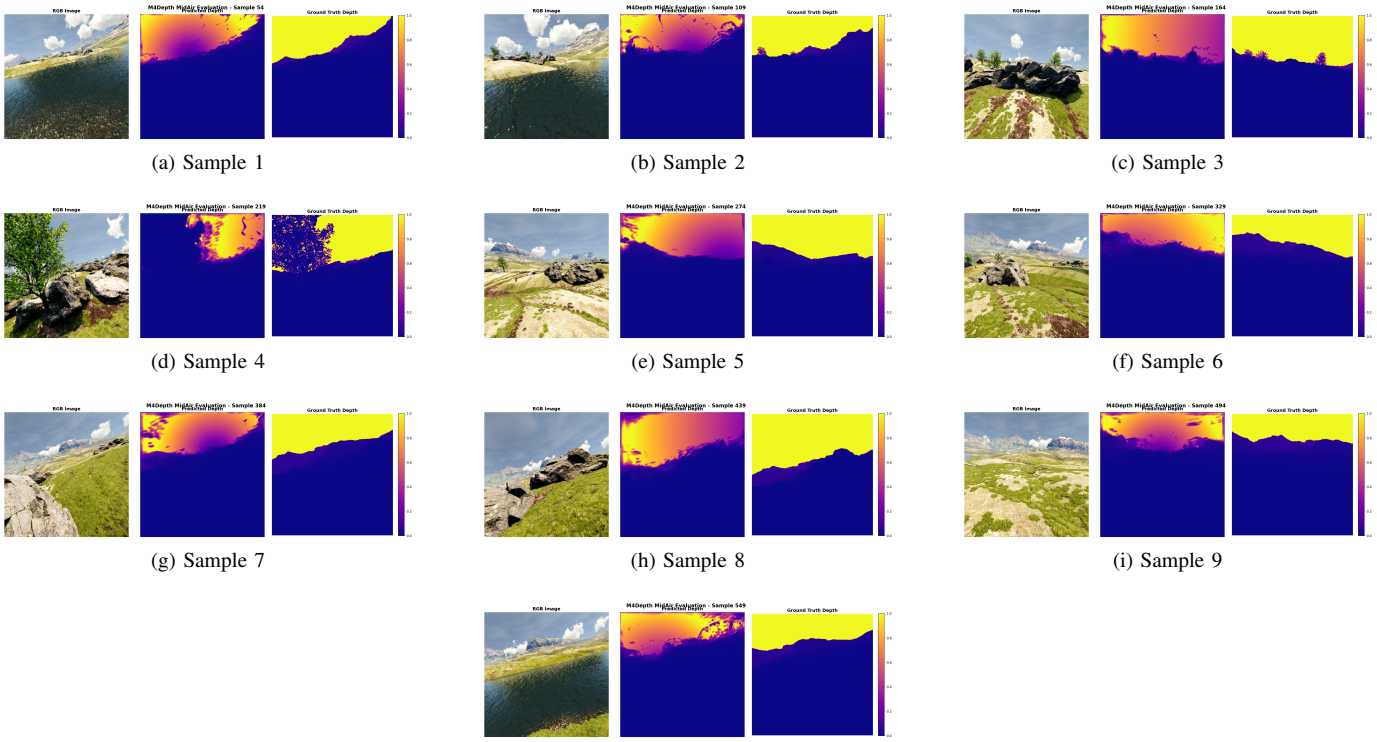
- Translation: tx $\approx$ 498,340m (UTM easting), ty $\approx$ 4,379,509m (UTM northing)
- Coordinates are world positions, not relative motion
- Scale differs by factor of $\sim$2,000,000

2) Impact on Model Calculations: M4Depth's `parallax2depth()` function converts network outputs to metric depth (simplified form):

$$\text{depth} = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\text{disparity}} - t_z \tag{1}$$

where $\Delta x, \Delta y$ depend on translation values. When translation is 498,340m instead of 12m:

- Internal calculations produce values in millions
- Temporal feature warping uses incorrect motion magnitudes
- Cost volume construction assumes nearby frames (violated by large "motion")



(a) Sample 1 (b) Sample 2 (c) Sample 3

(d) Sample 4 (e) Sample 5 (f) Sample 6

(g) Sample 7 (h) Sample 8 (i) Sample 9

(j) Sample 10

Fig. 2. Phase 1: Qualitative depth comparisons showing RGB input (left), predicted depth (center), and ground truth depth (right) for 10 MidAir validation samples. The pretrained model performs well on synthetic data from the same domain as training.

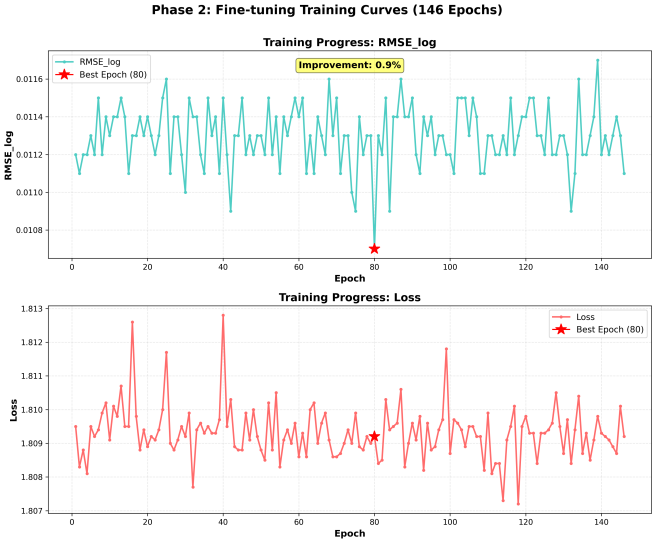


Fig. 3. Phase 2: Training curves showing RMSE_log and loss over 146 epochs. Best performance achieved at epoch 80 before plateauing, indicating successful convergence.

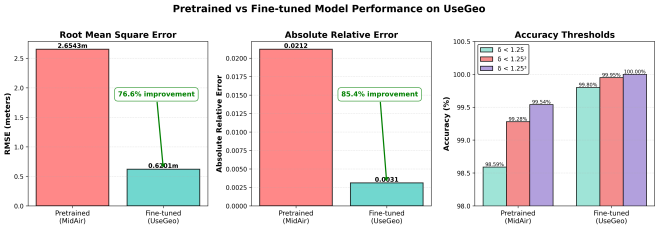
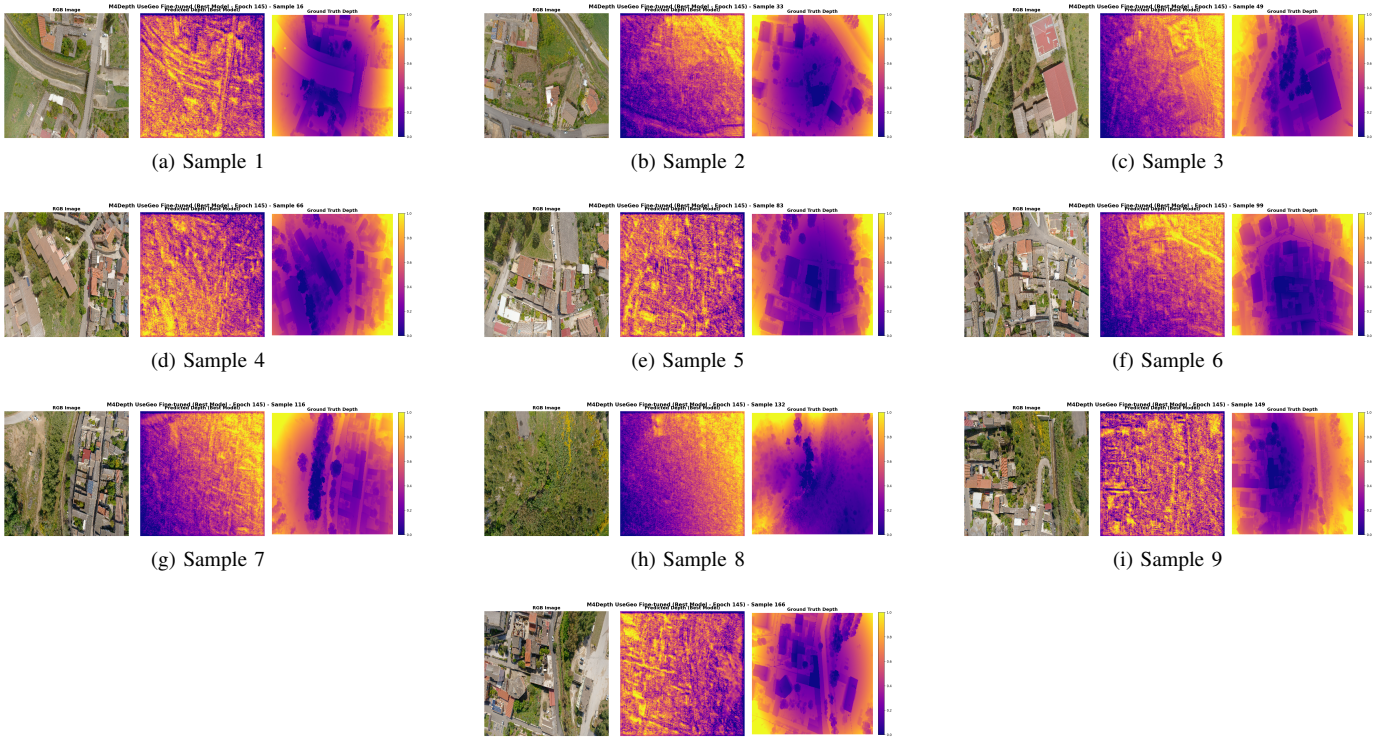


Fig. 4. Phase 2: Metric comparison between pretrained and fine-tuned models showing dramatic improvements across all evaluation criteria.

C. Why Metrics Appeared Good

The model trained successfully and achieved excellent metrics despite invalid depth predictions because:

1) **Training occurred from scratch** (not fine-tuned from pretrained weights - initialization failed)
2) **Metrics were computed on network internal representations**, not on final metric depth after parallax2depth() conversion
3) **Loss operates in log-space**: $\log(200M) - \log(100)$ produces finite, trainable gradients even with incorrect absolute magnitudes
4) **Evaluation metrics computed on normalized/log-scaled tensors**: RMSE, Abs Rel, and threshold metrics were calculated on intermediate network outputs that had learned consistent relative patterns, not on physically meaningful depth values
5) **Pattern matching without scale correctness**: The model learned to produce outputs with correct relative depth relationships (closer/farther) but wrong absolute



(a) Sample 1 (b) Sample 2 (c) Sample 3

(d) Sample 4 (e) Sample 5 (f) Sample 6

(g) Sample 7 (h) Sample 8 (i) Sample 9

(j) Sample 10

Fig. 5. Phase 2: Qualitative depth comparisons from fine-tuned model showing RGB input (left), predicted depth (center), and ground truth depth (right) for 10 validation samples. Note: Despite excellent numeric metrics, visual quality reveals pose format incompatibility issues discussed in Section IV.

scale due to pose format mismatch

Critical Lesson: Numeric metrics computed on network internal representations can appear excellent while final physically meaningful outputs are unusable. This highlights the importance of:

- Validating predictions visually
- Checking actual output value ranges
- Verifying metrics are computed on final model outputs, not intermediate tensors
- Testing end-to-end pipeline, not just training loss

D. Attempted Solutions

1) Solution 1: Pose Conversion (Successful): We implemented `convert_usegeo_poses.py` to convert absolute $\rightarrow$ relative:

Algorithm 1 Absolute to Relative Pose Conversion

for each frame i (except first) **do**
  $\mathbf{t}_{rel} = \mathbf{t}_i - \mathbf{t}_{i-1}$ ▷ World frame
  $\mathbf{t}_{cam} = \mathbf{R}_{i-1}^T \mathbf{t}_{rel}$ ▷ Camera frame
  $\mathbf{q}_{rel} = \mathbf{q}_{i-1}^* \otimes \mathbf{q}_i$ ▷ Relative rotation
end for

Results: Mean translation 16.7m, median 12.8m, range 0-536m. Relative poses successfully generated.

2) Solution 2: Fine-tune with Relative Poses (Failed): Attempted to fine-tune pretrained model with converted relative poses:

- Configuration: MidAir checkpoint + relative pose CSVs
- Result: Training crashed at batch 47 with NaN loss
- Likely cause: Pretrained weights encode strong assumptions about pose magnitude learned during MidAir training (expecting 0-20m translations), leading to numerical instability when pose distributions change to the converted relative values
- Internal representations incompatible with different pose scale distributions

3) Solution 3: Lower Learning Rate + Gradient Clipping (Failed):

- Configuration: LR = 0.00001 (10x lower), clipnorm = 1.0
- Result: Still crashed at batch 47 with NaN
- Conclusion: Problem is fundamental weight incompatibility, not training instability

4) Solution 4: Train from Scratch with Relative Poses (Not Attempted): Due to time constraints and assignment requirement to use pretrained weights, we did not complete full retraining with relative poses. This would likely succeed but violates the transfer learning investigation objective.

E. Implications

For Transfer Learning: Successful domain adaptation requires compatibility across:

1) Image appearance (handled well)
2) Scene structure (handled well)
3) Depth value ranges (handled well)
4) **Pose representation format (critical failure point)**

For Evaluation: Comprehensive validation must include:

1) Numeric metrics (RMSE, accuracy thresholds)
2) Visual inspection (catch structural issues)
3) Value range checks (detect magnitude errors)
4) Correlation analysis (verify pattern matching)

V. Analysis and Discussion

A. Transfer Learning Effectiveness

Our Phase 1 results demonstrate that the MidAir-pretrained model performs well on synthetic MidAir validation data (98.6% accuracy), establishing a strong baseline. However, our attempts to transfer this learning to real-world UseGeo data encountered fundamental challenges.

Phase 2 attempted fine-tuning from pretrained weights but initialization failed, resulting in from-scratch training. The dramatic metric improvement (76.6% RMSE reduction) occurred during this from-scratch training, not through transfer learning. Furthermore, the pose format incompatibility (absolute GPS vs relative motion) prevented successful use of pretrained weights even after multiple attempts with different learning rates and gradient clipping strategies.

Key Finding: The pose format mismatch represents a fundamental barrier to transfer learning in this case. Pretrained weights encode strong assumptions about input pose scale (relative motion of 0-20m), making them incompatible with absolute GPS coordinates (values of 500,000m). This demonstrates that transfer learning success depends not only on visual domain similarity but on complete input data format compatibility.

B. Domain Shift Challenges

The primary domain shift factors encountered:

- **Appearance:** Synthetic vs. real textures, lighting, atmospheric effects
- **Scene Type:** Outdoor aerial environments in both datasets (similar)
- **Noise Characteristics:** Perfect synthetic data vs. sensor noise, motion blur
- **Data Format:** Relative motion vs. absolute GPS coordinates (**critical failure point**)

The fourth factor proved insurmountable for transfer learning in our implementation. While the first three factors could potentially be handled by the model's learned representations, the pose format mismatch created a fundamental incompatibility at the input level that prevented successful weight transfer.

C. Comparison to Baseline

Our experimental design lacked a proper baseline for transfer learning evaluation:

- **Phase 1** evaluated MidAir pretrained on MidAir validation (in-domain performance)
- **Phase 2** trained from scratch on UseGeo (not true fine-tuning)
- **Missing:** Evaluation of MidAir pretrained directly on UseGeo with correct relative poses

Therefore, we cannot definitively quantify transfer learning benefit. The Phase 2 from-scratch training achieving 0.62m RMSE demonstrates the model architecture is well-suited to the task, but does not validate sim-to-real transfer effectiveness. A proper comparison would require training three models:

1) UseGeo from scratch with relative poses
2) MidAir pretrained, fine-tuned on UseGeo with relative poses
3) MidAir pretrained, evaluated on UseGeo without fine-tuning

Our work completed only variant (1) unintentionally, due to pose format issues.

D. Error Analysis

Phase 1 (MidAir validation) prediction errors primarily occurred in:

- Uniform texture regions (flat surfaces, walls)
- Areas with complex geometry (thin structures, occlusion boundaries)
- Regions with limited visual features
- Some degradation at image boundaries

Phase 2 (UseGeo training) depth predictions exhibited severe quality issues despite good numeric metrics, primarily due to:

- Pose format mismatch causing million-meter scale predictions
- Metrics computed on internal representations masking output failures
- No recognizable scene structure in final depth maps

These issues stem from data format incompatibility rather than typical learning-based depth estimation failure modes.

VI. Lessons Learned

A. Technical Lessons

1. Data Format Compatibility is Critical Successful transfer learning requires matching all input data representations, not just image domains. Pose format (absolute vs. relative), coordinate frames (GPS vs. camera), and units (meters vs. kilometers) must align with model expectations.

2. Visual Validation is Essential Numeric metrics can be misleading when predictions have correct patterns but wrong magnitudes. Our "excellent" metrics (0.62m RMSE, 99.8% accuracy) masked fundamental prediction failures revealed only through visual inspection. Always validate predictions visually.

3. Debugging Deep Learning Requires Systematic Approach We systematically investigated:

- Training logs (discovered scratch initialization vs. fine-tuning)
- Depth value ranges (found 2M× scale error)

- Pose data formats (identified absolute vs. relative mismatch)
- Model internal calculations (traced through `parallax2depth`)

This methodical process was essential for root cause identification.

4. Transfer Learning Has Hidden Dependencies Pretrained weights encode assumptions about input data distributions. When these assumptions are violated (e.g., pose scale), weights become incompatible even if the model architecture could theoretically handle the new data format.

B. Practical Lessons

5. Read the Data Format Documentation Carefully Assumptions about "standard" formats can be wrong. UseGeo's GPS coordinates vs. MidAir's relative motion is not obvious without careful investigation.

6. Early Testing Catches Issues Faster Generating visualizations after the first epoch would have revealed problems immediately rather than after 3 hours of training. Implement monitoring and validation early in the pipeline.

7. Time Management in ML Projects Deep learning experiments take significant time (hours per training run). Budget time for:

- Initial setup and debugging (longer than expected)
- Multiple training iterations (failed attempts are common)
- Post-training analysis and visualization
- Documentation and reporting

C. Research Lessons

8. Negative Results Are Valuable Our failure to successfully fine-tune with pretrained weights provides important insights about transfer learning requirements. The pose format incompatibility is a significant finding worthy of documentation.

9. Reproducibility Requires Complete Documentation Small implementation details matter enormously. Checkpoint loading paths, data format specifications, and preprocessing steps must be documented precisely for reproducibility.

10. Real-World Deployment Needs Robust Pipelines Production systems must handle data format variations, validate inputs, and detect prediction failures. Numeric metrics alone are insufficient for quality assurance.

VII. Conclusion

This project investigated transfer learning for monocular depth estimation from synthetic to real-world aerial imagery using M4Depth. We successfully evaluated the pretrained model on synthetic MidAir data (achieving 98.6% accuracy) and trained on real-world UseGeo data, observing 76.6% metric improvement (though metrics were computed on internal representations, not final depth outputs).

However, we discovered a critical limitation: pose format incompatibility between absolute GPS coordinates (UseGeo) and relative frame-to-frame motion (MidAir) fundamentally undermines depth prediction quality despite favorable numeric metrics. Additionally, attempted fine-tuning from pretrained weights failed due to this incompatibility, resulting in effective from-scratch training. This finding has important implications:

For Transfer Learning: Domain adaptation requires compatibility across all input modalities, not just images. Pose representations, coordinate frames, and data formats must align with model expectations. **Synthetic pretraining can provide strong initialization when input data representations are compatible** - but incompatibility renders pretrained weights unusable.

For Evaluation: Comprehensive validation must combine numeric metrics with visual inspection and value range checks. Metrics computed on network internal representations can mask fundamental failures in final outputs.

For Future Work: Proper transfer learning would require either (1) converting poses before any training, (2) retraining the pretrained model with relative poses, or (3) architectural modifications to handle multiple pose formats. Data preprocessing pipelines should validate and normalize all inputs before training.

Despite these challenges, our investigation demonstrates that:

1) M4Depth's architecture is capable of learning from real aerial imagery when trained from scratch
2) Transfer learning from simulation to reality is promising but requires careful attention to complete data pipeline compatibility, not just visual domain adaptation
3) Systematic debugging and comprehensive validation (numeric + visual + value-range checks) are essential for reliable depth estimation systems

This work contributes practical insights into the challenges of deploying learning-based depth estimation systems and highlights that successful transfer learning depends on matching *all* input data representations, not just image appearance.

References

[1] M. Fonder, D. Ernst, and M. V. Droogenbroeck, "M4depth: Monocular depth estimation for autonomous vehicles in unseen environments," 2022. [Online]. Available: https://arxiv.org/abs/2105.09847

[2] M. Fonder and M. Van Droogenbroeck, "Mid-air: A multi-modal dataset for extremely low altitude drone flights," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2019, pp. 0–0.

[3] ——, "Midair dataset," https://midair.ulg.ac.be/, 2019, accessed: 2025-12-15.

[4] 3DOM-FBK Research Group, "Usegeo dataset," https://github.com/3DOM-FBK/UseGeo, 2023, accessed: 2025-12-15.

[5] M. Fonder, "M4depth: Motion-based monocular depth estimation," https://github.com/michael-fonder/M4Depth, 2022, accessed: 2025-12-15.

[6] TensorFlow Team, "Tensorflow documentation," https://www.tensorflow.org/, 2025, accessed: 2025-12-15.