Structured modeling of LiDAR point clouds

Michael Stanley
NYU Center for Data Science

November 11, 2020

Aerial LiDAR Scanning

LiDAR

- Use of laser pulses to measure distances
- LiDAR scanner **emits laser pulses**, measures the time required for the pulse to hit an object, return to scanner
- Point location is calculated based on **return time** and **scanner orientation**, then **georeferenced** based on location of scanner

Aerial LiDAR

- LiDAR scanner attached to an **aircraft**
- Invented in the 1960s, aerial LiDAR was used primarily in **elevation/topological mapping**, used more frequently in urban environments in recent years

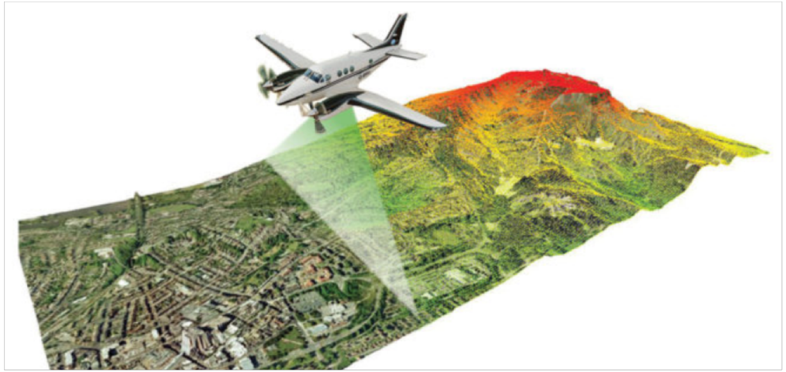


Fig: Aerial LiDAR Scanning[1]

Fig: Resulting point cloud from 2019 scan of Sunset Park, Brooklyn.

Point Clouds

- Aerial LiDAR scanning creates a **point cloud**
- Each point has **three spatial coordinates**, other features (intensity, return number, etc.)

[1]*https://www.auav.com.au/articles/drone-data-vs-lidar/*

Point Cloud Processing

- **Applying machine learning** to point clouds is problematic for a few reasons:
  - 3-dimensional
  - Unordered
  - Density varies (especially in aerial LiDAR)
- In aerial LiDAR, many ML approaches **project the point** cloud onto a 2-D plane, apply image processing
  - Successful in some contexts, but projection **discards the geometry** of the scene

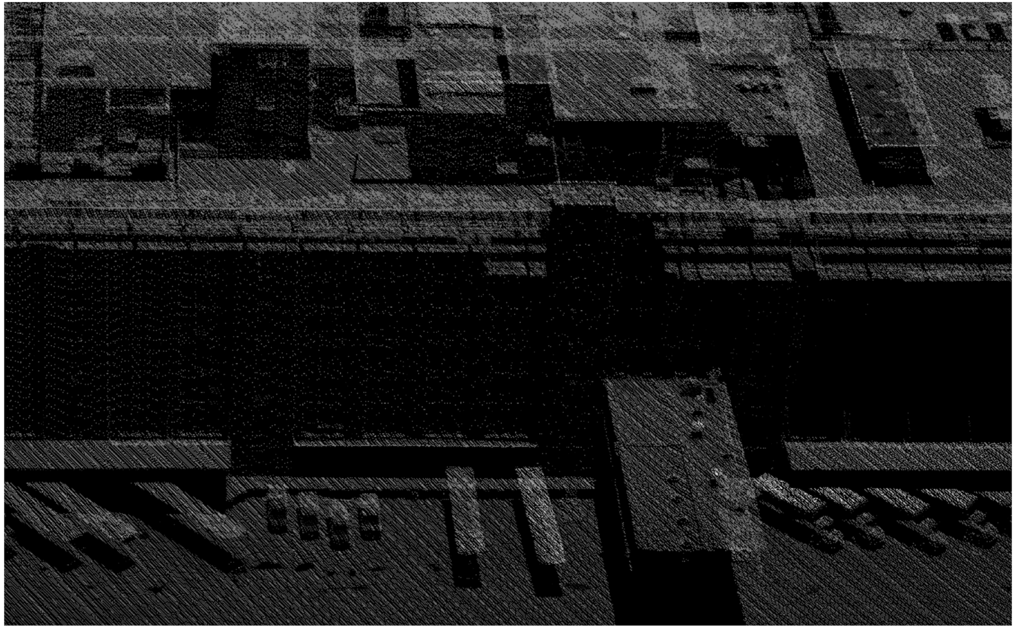


Fig: Whiter areas indicate greater point density, darker areas indicate low density

- More recently, **PointNet**[1] and related approaches **process the point cloud directly**
  - Performs well for **object detection** and **classification**
  - Has not been applied to **regression**, **inverse problems**

[1] Qi et al. (2017). *Pointnet: Deep learning on point sets for 3d classification and segmentation.*

Structured LiDAR Collection

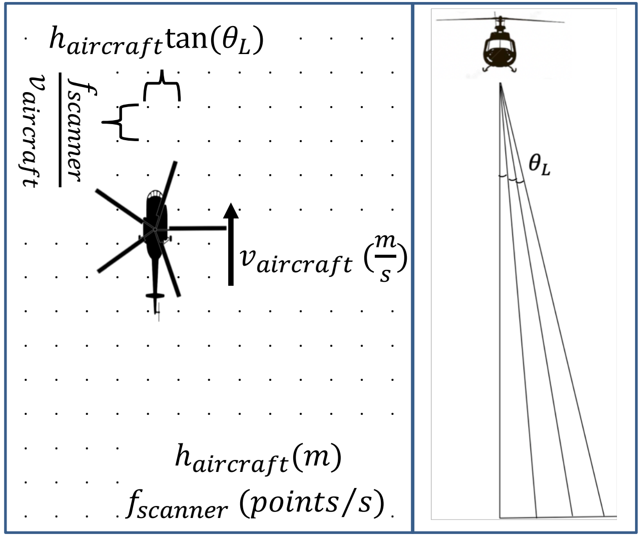


Fig. Flight parameter impact on point cloud grid

- LiDAR point clouds are **unordered**, but they are **collected in a highly structured** manner
    - LiDAR scanners use **constant scan angles** and **scan frequency**
    - Aircraft fly at **constant speed**
- Point clouds generated from a single aerial flight pass approximate **a grid**.
- Caveats
    - **Grid spacing** may not be equal in both directions, or constant along the scan line
    - The grid does not map directly to x and y coordinates in the scanned environment
    - We treat the **spatial coordinates as features** of each point on the grid

Problem: Missing Point Inpainting

- If a pulse does not return to the scanner, a **point is missing**
  - Missing points are identifiable where gaps in the scan angle between consecutive points are too large
- 3-5% of points in the Brooklyn dataset are missing
- Causes of missing points
  - Water
  - Rough surfaces
  - Scanner failure
- **Problem Statement**: Can we utilize the structure of a point cloud to inpaint (i.e. fill) these missing points via machine learning?

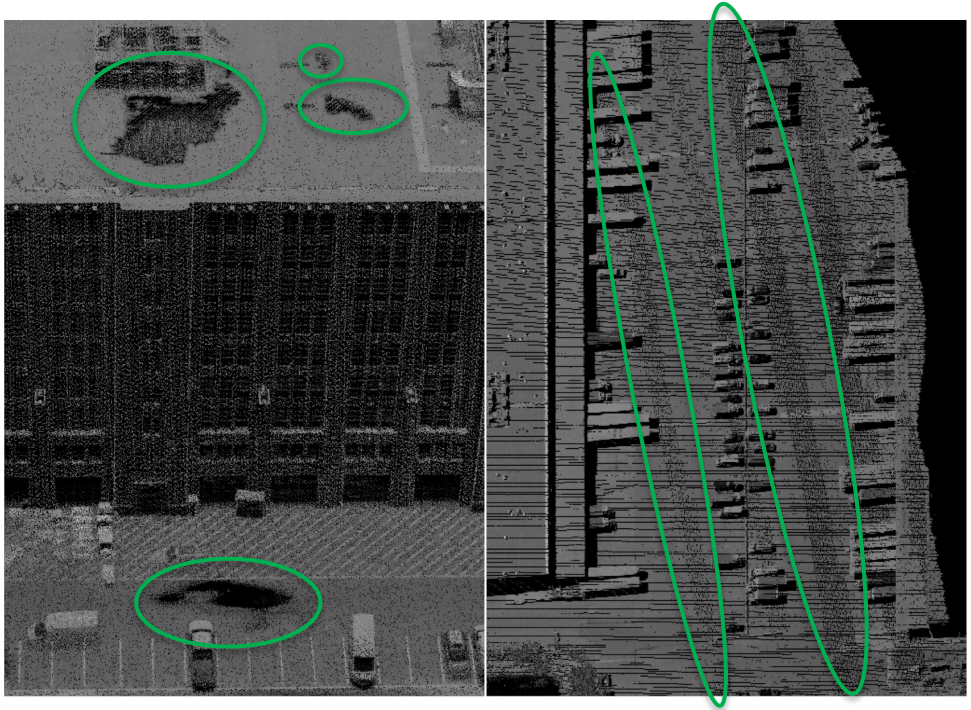


Fig. Missing points due to standing water (left) and scanner failure (right)

2 modeling approaches

1-D Sequence

- Each scan line is a **sequence of points**
- Apply sequence models to predict x, y, and z coordinates of missing points
  - Recurrent Nets, 1-D Convolutional Nets, Transformers
- *Model used: 1-D CNN, U-Net architecture*

2-D Grid

- Consecutive scan lines form **a 2-D grid** of points
- Scan lines aligned by scan angle
- Apply image models to predict x, y, and z coordinates of missing points
  - 2-D Convolutional Nets
- *Model used: 2-D CNN, U-Net architecture*

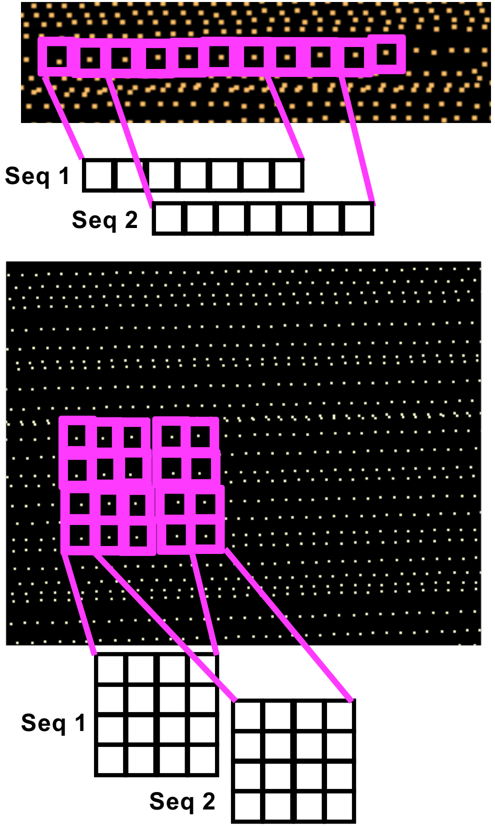


Fig. Generating 1-D (top) and 2-D (bottom) samples from point cloud

Model architecture

U-Net

- We use the U-Net architecture for both **1-D and 2-D** models
  - Commonly used architecture, features a contracting path followed by an expanding path, ”U” shaped.
  - **Wide receptive field** allows far away points in sequence to impact prediction.
  - Applicable to different input sizes, dimensions

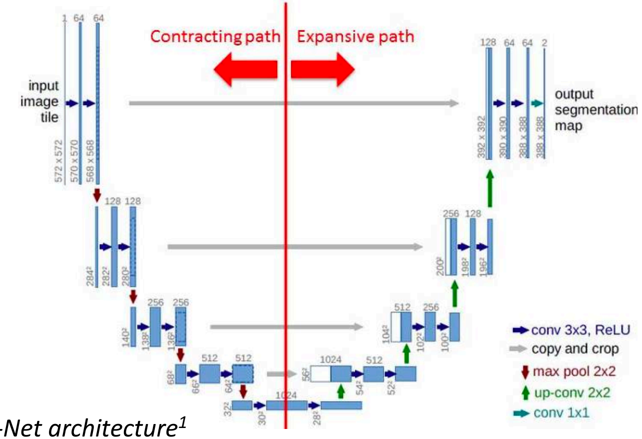


Fig: U-Net architecture[1]

PartialConv

Original Image **Mask** **Inpainted image**

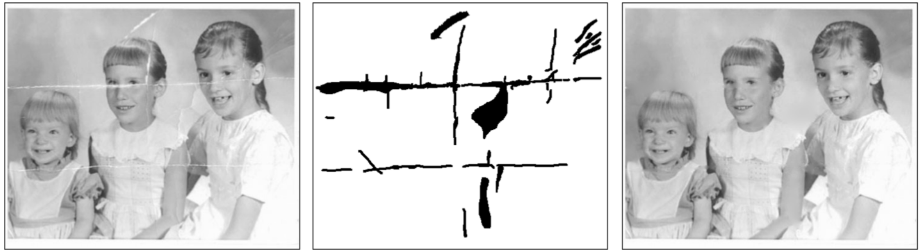


Fig: Image and Mask input example[3]

- To indicate the missing points, we use partial convolutional layers[2] that accept an image and a mask
- Mask is reshaped and passed through the model
  - Each layer is aware of the mask

[1]Ronneberger et al. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation.*
[2]Liu et al. (2018). *Image inpainting for irregular holes using partial convolutions.*
[3]Oliveira et al. (2001) *Fast Digital Image Inpainting*.

Data

NYU TANDON SCHOOL OF ENGINEERING

Dataset

2019 aerial LiDAR scan of Sunset Park, Brooklyn

- 82 flight passes, each 50 seconds and ~12 million points
- Features
  - 3 spatial coordinates
  - Intensity
  - Scan angle, abs(scan angle)
  - Scan line number

Sampling

- **1-D**: 256-point sequences
  - 24,000 training samples, 6,020 validation samples
- **2-D:** 32x32-point squares
  - 38,000 training samples, 9,000 validation samples
- Samples discarded if contain actual missing points

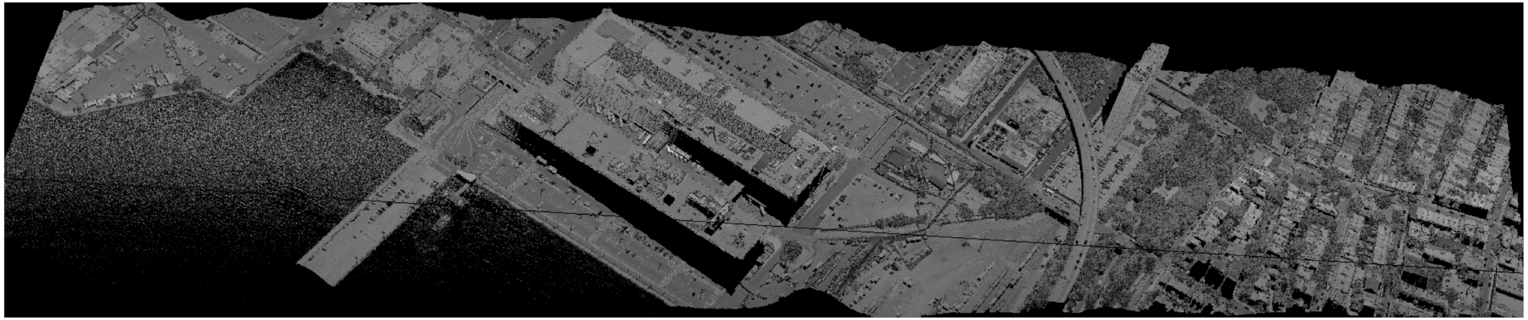


Fig: Flight strip 164239 from Sunset Park 2019 scan, colored by intensity.

Initial Results

	Training Loss	Validation Loss
1-D Sequence		
Baseline	1.668	**2.157**
U-Net w/ PartialConv	**1.609**	9.128
2-D Grid		
Baseline	1.522	2.706
U-Net w/ PartialConv	**0.329**	**0.661**

- Both 1-D and 2-D models appear to have the capacity to learn the shapes of the scanned environment
- 1-D model is **overfitting** the training data
- 2-D model substantially outperforms the baseline in both training and validation
  - 2-D infill is a much harder task than 1-D, so the baseline is not as successful

Note: 1-D and 2-D use different training/validation sets

MSE Loss

$$l_{1D}(y,\hat{y}) = \frac{1}{n*m}\sum_{i=1}^{n}\|y_i - \hat{y}_i\|_2^2$$

$$y_i, \hat{y}_i \in \mathbb{R}^d$$

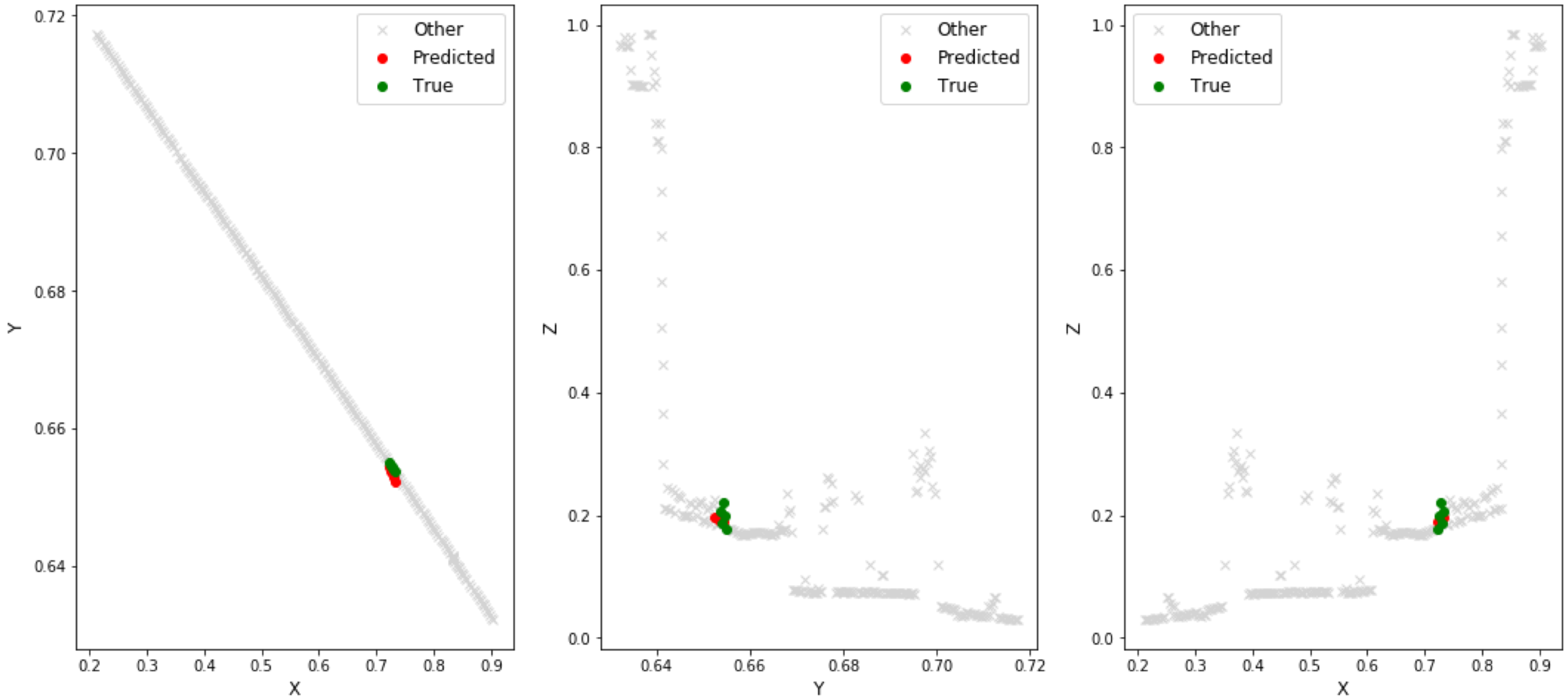
$$l_{2D}(Y,\hat{Y}) = \frac{1}{n*m}\sum_{i=1}^{n}\|Y_i - \hat{Y}_i\|_F^2$$

$$Y_i, \hat{Y}_i \in \mathbb{R}^{dxd}$$

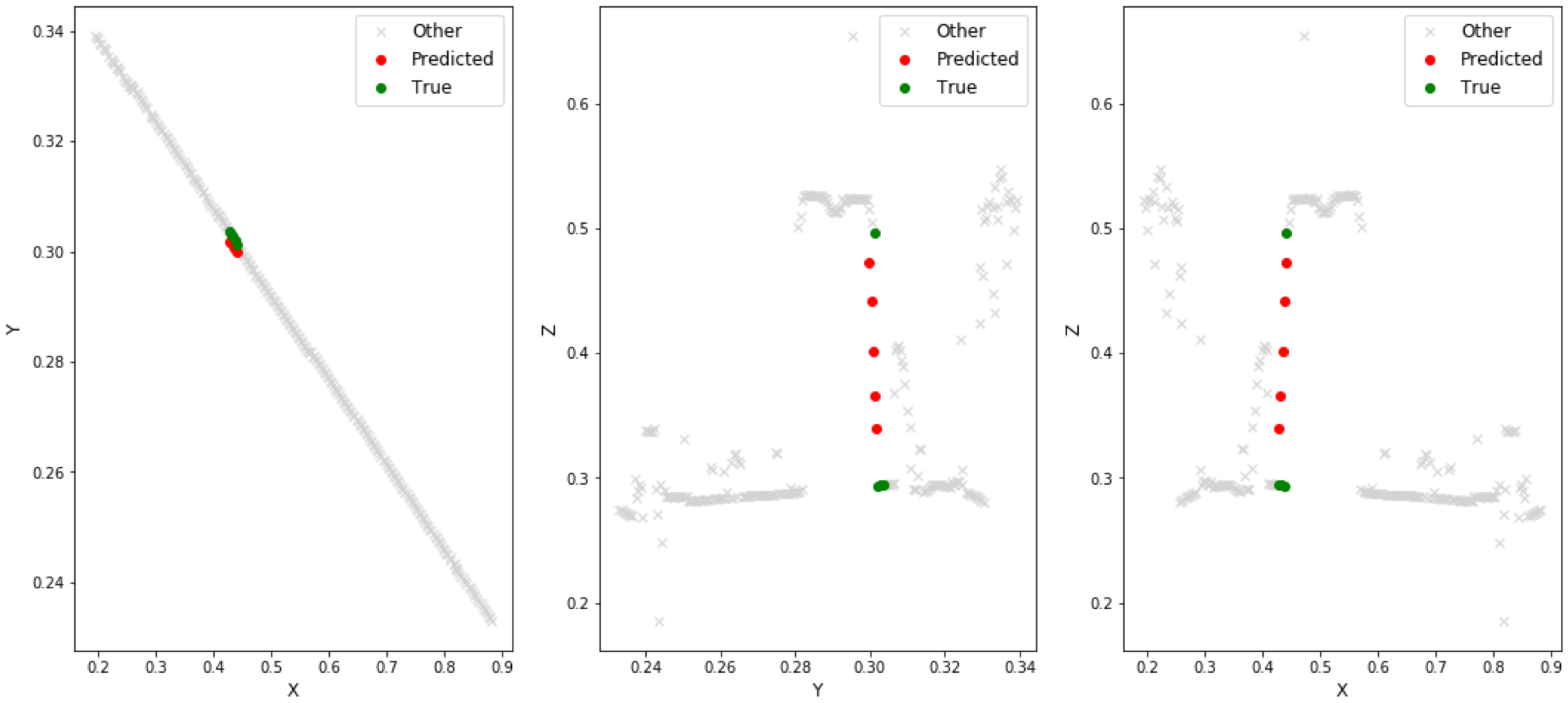
n – Number of samples

m – Number of masked points per sample

1-D Model: Training Set

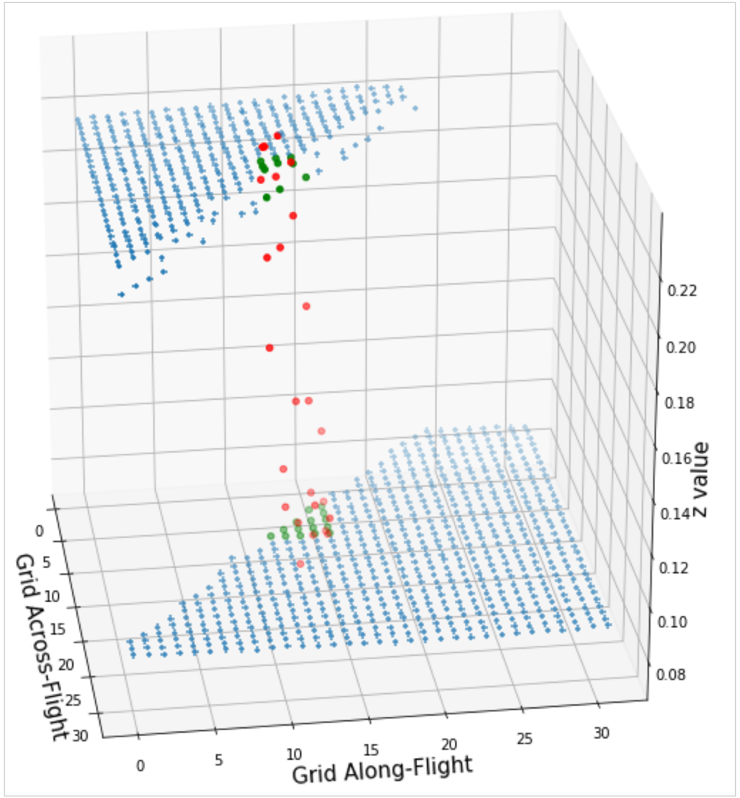


1-D Model: Validation Set

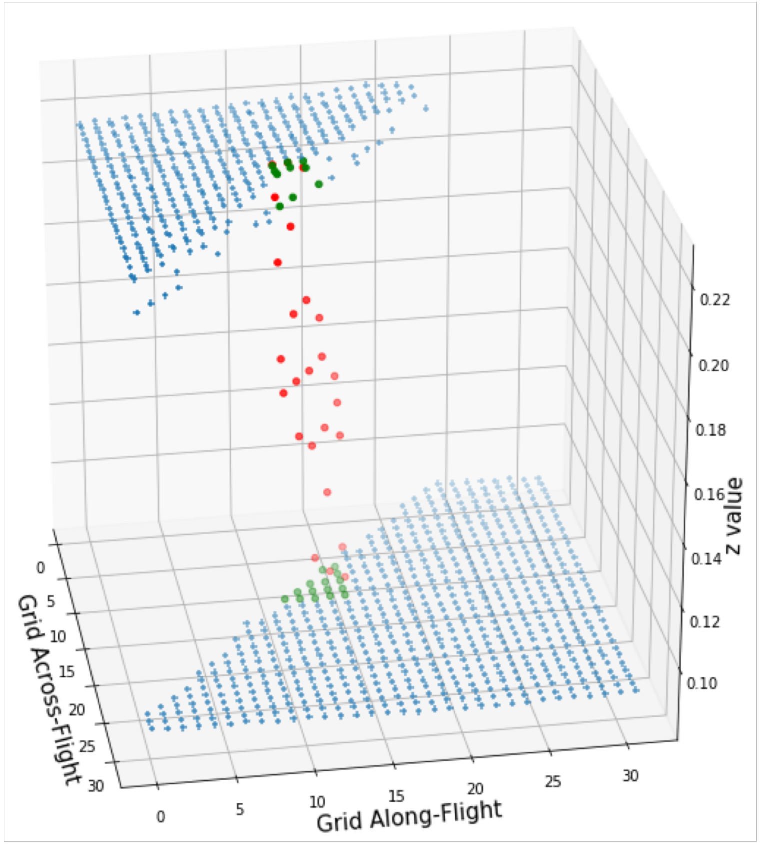


2-D Model: Training Set

Model

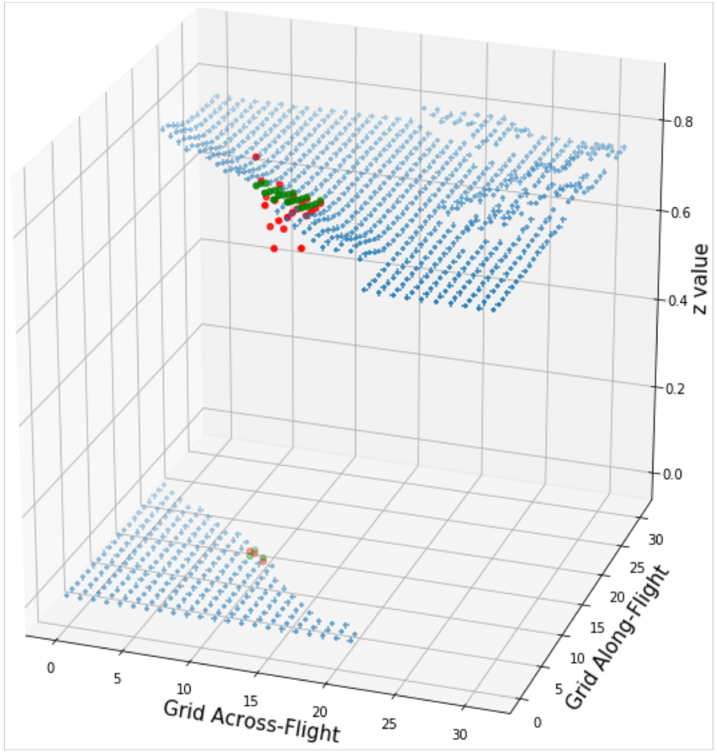


Baseline

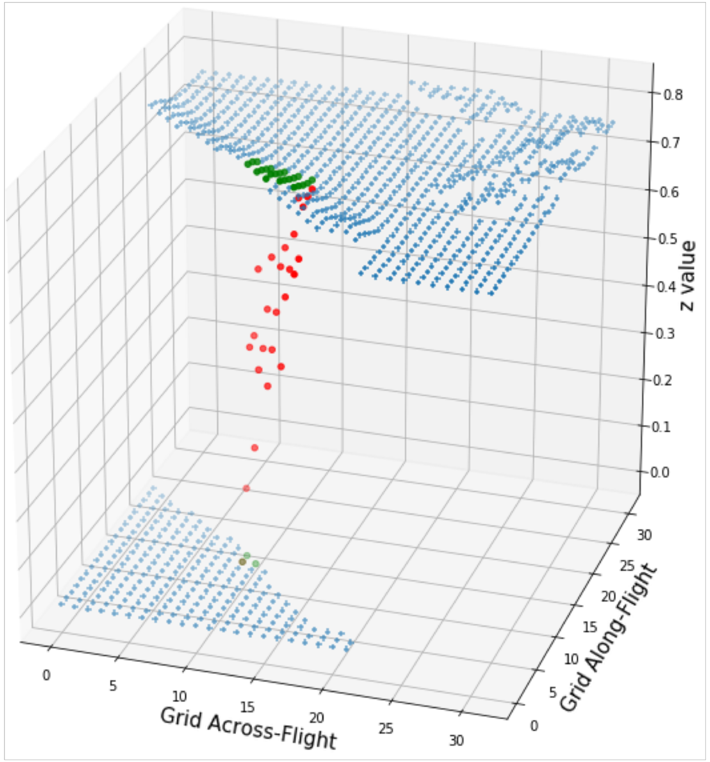


2-D Model: Validation Set

Model



Baseline



Conclusion

NYU TANDON SCHOOL OF ENGINEERING

Next Steps

- Evaluate different **sample sizes**, model **architectures**
- Do models **generalize** to different
  - mask sizes?
  - flight passes?
  - datasets?
- Where does the model perform well? Does this align with regions of interest (e.g., **vertical surfaces**)?

Extensions

- Similar approaches should extend beyond filling missing points to other problems
  - Super resolution
  - Denoising
- Feature extraction per flight pass, could feed into downstream processing of the full point cloud

NYU TANDON SCHOOL OF ENGINEERING

Thank you!

Appendix

Baseline & Loss Measure

Baseline

- Compare models to a non-learning, deterministic approach
- **1-D**
  - Interpolation based on nearest, unmasked points
- **2-D**
  - Fit a plane to neighboring, unmasked points, inpainted value is on the plane
  - Iterative from border to center of masked region

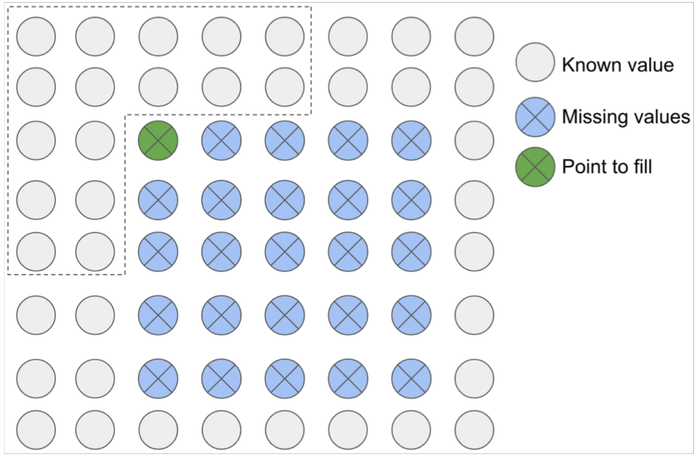


Fig: 2-D baseline approach, using n=2 (neighbor distance)

Loss Model

- Weighted MSE Loss
  - Accounts for scale of spatial dimensions
  - Data is otherwised normalized
- Otherwise, model does not learn precise predictions for x and y coordinates, as they have much larger range in the data than z