SynthCave: A Deep Learning Benchmark for 3D Odometry Estimation in Caves

Tim Alexander Bader
tim.bader@uni-ulm.de
Universität Ulm
Ulm, Germany

ABSTRACT

Accurate position and odometry estimations are essential for autonomous navigation in indoor environments where GPS signals are often unavailable. Current methods rely on sensors such as cameras, IMUs, and LiDARs, with vision-based systems being the most commonly used. However, in low-light environments, cameras become less effective. Furthermore, as far as we know, there are currently no publicly available 3D odometry estimation datasets. To find new solutions, we present SynthCave, the first 3D odometry estimation benchmark dataset that utilizes cave-like environments to amplify the challenges of indoor navigation. SynthCave comprises LiDAR data in three forms: point clouds, depth images, and graphs, along with IMU and ground truth data. Baseline results for SynthCave are presented using methods such as ZERO, RANDOM and CNNs as well as SOTA models such as TSViT, ASTGCN, and PSTNet, which are based on transformers, graph neural networks, and point networks, respectively. The experiments demonstrate that in most cases, the SOTA models outperform the well-established methods. Additionally, graph-based models are competitive with popular vision-based models for 3D odometry estimation. The impact of data representation and input size on the models' performance is also analyzed. The code[1] and the dataset[2] are publicly available.

KEYWORDS

Dataset, Benchmark, 3D Odometry Estimation, Graph Neural Networks, Computer Vision, IMU, LiDAR

1 INTRODUCTION

Position- and odometry estimation (PE, OE) have undergone significant advancements with the widespread availability of Global Positioning Systems (GPS), making it a relatively straightforward task. However, in indoor settings where GPS signals are inaccessible, achieving precise PE can present a formidable challenge. Alternative methods that use WiFi signal strengths [48] or fuse data from cameras, Inertial Measurement Units (IMU), and Light Detection and Ranging (LiDAR) sensors [38] offer a potential solution, yet they come with inherent drawbacks such as high costs and increased computational complexity, both of which can be limiting in certain cases. Additionally, environmental conditions may render specific sensors unusable, such as cameras in poorly lit buildings, tunnels or cave systems. Consequently, there is a need to explore novel approaches that address these limitations.
In recent years, Deep Learning (DL) systems have demonstrated remarkable effectiveness in predicting the odometry from various structured multi-modal inputs, often in the context of Simultaneous Localization and Mapping (SLAM) [38]. While the current DL focus in this area predominantly leans towards computer-vision-based systems, other approaches like graph-based ones are underrepresented. This underrepresentation may stem from the perception that graph-based approaches are more suited to unstructured data. However, recent research [13, 33] has demonstrated that graph-based methods can achieve state-of-the-art performance on structured data such as LiDAR point clouds. This opens up new avenues for exploration and underscores the potential of graph-based approaches in this domain.
Transitioning to the challenge of data availability, a primary limitation for DL systems arises from the substantial demand for training datasets, which are often difficult to obtain. Numerous extensive datasets [2, 45] and benchmarks [9] are available for outdoor PE and OE, with GPS serving as the primary ground-truth (GT) source. However, the scenario changes significantly when considering indoor settings. Only a limited number of datasets [1, 12, 18, 35, 37] using LiDARs are publicly available, and to our knowledge, a substantial indoor PE or OE dataset in a 3D space is currently non-existent. This lack of data underscores a significant gap in contemporary research and development efforts, potentially due to the absence of a large-scale approach for obtaining the GT 3D OE within indoor environments and the current focus on easy-accessible public buildings.
Our approach seeks to address this gap by transitioning to synthetic data derived from simulated environments, offering a promising alternative to retrieve the GT. However, this approach comes with a notable limitation: simulated data maintains perfect accuracy, often diverging from the inherent noise and unpredictable irregularities of real-world data. This limitation can be confronted by applying SOTA techniques like domain randomization [25, 40]. By exploring the usage of graphs in this context, we aim to provide a novel perspective and contribute to bridging this gap. This leads us to the following contributions:

(1) We propose the benchmark dataset SynthCave for 3D odometry estimation in cave-like environments. The LiDAR data is presented in three distinct forms: a) as point clouds, b) through depth-images, and c) in the form of graphs. All representations are complemented by IMU and GT data.
(2) We introduce baselines for SynthCave based on ZERO, RANDOM and CNN and explore the application of the SOTA models TSViT, ASTGCN and PSTNet.

Below is a summary of the related work. Section 3 provides a clear description of the composition of SynthCave. We introduce baseline models in Section 4 and describe their application to SynthCave in Section 5. We then present results in Section 6 and discuss them in Section 7 before coming to a conclusion.

2 RELATED WORK

Historically, LiDAR scan matching has played an important role in OE. Common scan matching algorithms are Iterative Closest Point (ICP) [3] and Polar Scan Matching (PSM) [4]. However, both have difficulty dealing with noisy and large data sets. ICP attempts to iteratively minimize the difference between two point clouds by applying a transformation from which the movement

[1] https://github.com/BaderTim/SynthCave
[2] https://kaggle.com/datasets/badertim/synthcave-3d-odometry-estimation

between the two scans can be inferred. PSM matches points based on their orientation, offering greater efficiency and reduced inaccuracies, especially for larger differences. On the other hand, using only IMU data, various forms of Kalman filters [16, 24] have proven to be good for OE. However, their drift leads to increasingly inaccurate results over time.
Tabib et al. [36] survey caves autonomously with drones, feeding OE and depth camera images into a Gaussian mixture model to produce local occupancy grid maps. Their OE uses VINS-Mono [28], a visual inertial odometry framework that receives IMU and camera data and uses non-linear optimization-based methods such as sliding windows. They also integrate a cascaded proportional derivative controller with a nonlinear Luenberger observer to better handle external acceleration and torque disturbances [22]. However, the authors still report that the state estimator drifts over time and suggest incorporating depth observations in future experiments.

2.1 Deep Learning Approaches

The concepts of minimization and estimation problems also align with deep learning paradigms, where optimization and estimation goals often involve minimizing disparities or errors in parameterized systems. In this context, AlexNet [17], a convolutional neural network (CNN), has emerged as a pioneering approach to DL with its deep architecture using convolutional layers [8]. Specifically tailored for grid-like data such as images, CNNs use convolutional layers with learnable filters. This architecture enables automatic extraction and abstraction of intricate hierarchical features, making CNNs adept at capturing complex patterns and structures in visual data.
Nicolai et al. [23] use a CNN to predict the odometry of an indoor robot. The CNN processes successive 2D panoramic depth images projected from point clouds produced by a 3D LiDAR sensor. Although their OE performance is below average, the authors note that different combinations of parameters or integration of LSTMs [11] for time consideration could improve their results, which Walch et al. [44] confirm.
The replacement of CNNs with Vision Transformers (ViTs) [5, 19] may offer another angle of improvement, as Sun et al. [34] and Françani and Maximo [7] show in their research. ViTs use a different architecture based on self-attention mechanisms derived from transformer models [42] originally designed for sequential data such as natural language. They work by dividing an input image into patches, which are then linearly embedded to create sequences of tokens. These tokens are subject to self-attention mechanisms that allow each token to attend to all other tokens, capturing global dependencies within the image. When dealing with sequential images, Tarasiou et al. [39] introduce ViTs with a factorized temporo-spatial encoder.
Specifically tailored for point cloud data, PointNet [26] and its successor, PointNet++ [27], also stand as key contributions to this area of research. PointNet is able to directly process unordered points from a point cloud, allowing global feature extraction without relying on predefined structures by using shared multi-layer perceptrons. It achieves permutation invariance using symmetric functions, allowing consistent results regardless of point order. The aggregation of local features produces a global representation of the entire point cloud, facilitating tasks such as object classification or segmentation.
Zheng et al. [52] utilize the benefits of PointNet and propose the LodoNet architecture for the OE of objects. First, they apply a similar projection technique as Nicolai et al. [23] but instead concatenate the transformed images vertically. Second, they utilize a scale-invariant feature transformation [41] to identify matching keypoints between the two images and, next, apply a PointNet segmentation that discards inappropriate keypoints. Finally, the selected matching keypoints are used as inputs to rotation estimation and translation estimation modules, each using an instance of PointNet too. Another relevant model architecture based on PointNet is PSTNEt [6] that comes with spatio-temporal convolutions on point cloud sequences. This allows the consideration of features from previous point clouds in a timeline, which is important for OE.
Similar to PointNet, Graph Neural Networks (GNNs) [29] are also capable of processing unordered points from point clouds by treating the points as nodes within a graph. GNNs then update their node representations by aggregating information from neighboring nodes through iterative message passing, capturing relationships within the data. Graph Convolutional Networks (GCN) [14] improve GNNs by applying graph convolutions to extract hierarchical features from point clouds, considering local and global relationships. Attention Graph Neural Networks (Attention GNNs) [10, 43, 51] integrate attention mechanisms to selectively attend to crucial nodes or relationships, enabling a nuanced feature extraction and discernment of important features within the graph. When dealing with temporal graphs, extensions of GCNs [31, 32] and GNNs [46, 49] exist that incorporate time as an additional dimension.
Specialized Point Graph Neural Networks (Point GNNs) [33] consider points as nodes, efficiently modeling spatial dependencies in unstructured point clouds. This makes them adept at tasks like object recognition and segmentation within 3D environments.

2.2 Existing Datasets

Current indoor localization datasets containing LiDAR data mostly cover 2D scenarios in university buildings [12, 37, 44], small rooms in general [1] and malls [18, 35]. Lee et al. [18], who also provide a benchmark, utilize LiDAR SLAM and structure-from-motion (SFM) algorithms to receive GT values for their measurements from two SOTA Velodyne VLP-16 LiDARs, multiple cameras and a magnetic rotary position encoder.
Transitioning to synthetic data, the Gibson Environment [47] offers a large-scale approach for generating specifically tailored data for real-world perception of embodied agents, which includes OE. To bridge the reality-gap between synthetic and real data, the authors make use of their large amount of target-source data and employ joint spaces [30] through training forward- and backward-models. Another popular method is Domain Randomization [40] (DR), which is a technique that involves training the model on a variety of synthetic data generated by randomizing different parameters of the environment. This helps the model to generalize better to real-world data. Structured DR [25] is a variant of DR that involves randomizing the environment in a structured manner. Active DR [21] is another variant of DR that involves integrating DR into the learning process.

3 COMPOSITION OF SYNTHCAVE

In this Section, we give insights on how our SynthCave benchmark is composited. First, we explain the data collection process in Section 3.1. Second, we describe the post-processing methods in Section 3.2. Finally, we pour the finalized data into three different data representation types defined in Section 3.3, which we

Figure 1: Example LiDAR covering a 60°x60° area with a resolution of 240x240. In the left image, a cave section from Minecraft is shown. In the center image, the LiDAR rays are visualized. In the right image, a depth image from the LiDAR's distance measurements is plotted.

ultimately provide. We evaluate the benchmark with different baseline models defined in Section 4, according to the experimental apparatus in Section 5, and provide results in Section 6.

3.1 Data Collection

We collect the data for SynthCave by carrying out cave surveys in the simulated world environment of Minecraft [3] version 1.20.2. In addition to simply obtaining the GT odometry, Minecraft also brings advantages such as the procedural generation of caves with different sizes, shapes, features and biomes. Ingame, we apply the potions *Splash Potion of the Turtle Master* and *Splash Potion of Slow Falling* to our player, which reduces movement and falling speed. For simulating LiDAR and IMU sensors, we use the Minecraft Measurement Mod[4] with Minecraft Forge[5] version 48.0.35. We instantiate multiple sensors according to Table 1 and capture sequences from subjectively interesting cave sections as shown in Table 2. An example-setup is presented in Figure 1.
We formalize the data by defining a cave section measured over time as C, holding IMU scans $IMU_{C,t}$, LiDAR scans $S_{C,L,t}$ and GT values $GT_{C,t}$ with time step $t \in T$. A LiDAR $L \in LIDARS$ comes with a horizontal and vertical resolution $H_L \times V_L$ and executes distance scans defined as $D_{C,L,t} = (d_{1,1}, d_{1,2} ... d_{H_L,V_L})$, which are then transformed to a 3D scan $S_{C,L,t} = (p_{1,1}, p_{1,2} ... p_{H_L,V_L})$ with $p_{h,w} = \{x, y, z\}$. We consider just a single IMU, capturing the 3D linear and angular velocity with the attributes $IMU_{C,t} = \{accX, accY, accZ, rotX, rotY, rotZ\}$. Finally, the GT is defined as $GT_{C,t} = \{posX, posY, posZ, \phi, \theta\}$, where ϕ is the horizontal and θ vertical rotation in degrees. It contains the difference of the carrier's position and view angle to the previous frame from $t-1$.

Sensor Type	Freq. (Hz)	Coverage (°)	Resolution
Default LiDAR	5	120x40	240x80
Wide LiDAR	5	180x40	180x40
Angled LiDAR	5	120x80	120x80
IMU	10	-	-

Table 1: Sensors used in SynthCave. Every LiDAR has a range of 100m.

[3] https://minecraft.net/
[4] https://github.com/BaderTim/minecraft-measurement-mod
[5] https://forums.minecraftforge.net/

3.2 Post-Processing

We try to approximate real-world conditions of LiDAR and IMU sensors, targeting situations where scans could be imprecise. Example reasons for this are wrong calibration settings, technical faults, external magnetic fields, reflection behaviors of different materials and different volumes like water or dusty air.
A wrong calibration $CAL(C_{IMU})$ is influencing results from the first IMU scan $IMU_{C,1}$ through $IMU_{C,T}$. Technical faults $TF(C_L)$, like LiDARs not measuring properly at certain angles, are quite similar, but with the difference of possibly becoming defect after a certain time $def \in [1, T]$ has already passed. We do not consider IMU failures, but when looking at the inaccuracies $MAG(C_{IMU})$ caused by a single external magnetic field, the disturbance lies within $[mf_{start}, mf_{end}] \subset [1, T]$. We conclude that reflections caused by different terrain materials occur in areas and not just points, possibly resulting in inaccurate 3D patches $REF(C_L)$ in the 3D grid of $T \times H_L \times V_L$. A similar argumentation $VOL(C_L)$ can be used for different volumes like water. In conclusion, we define structured DR for our use case in Equation 1

$$\begin{aligned} DR_1(C_{LIDARS}) &= \oplus_{L \in LIDARS} VOL(REF(TF(C_L))) \\ DR_2(C_{IMU}) &= MAG(CAL(C_{IMU})) \\ DR(C) &= DR_1(C_{LIDARS}) \oplus DR_2(C_{IMU}) \end{aligned} \tag{1}$$

where C_{LIDARS} contains the scans produced by the LiDAR sensors and C_{IMU} the IMU scans and apply it to the collected data. Finally, we also convert the absolute GT values to the relative positional and rotational change compared to the previous frame. The histograms shown in Figure 2 and Figure 3 give insights into the distribution of the GT values. While the horizontal movement is quite equally distributed, the vertical movement shows a significant imbalance towards 0. When observing the rotations, an even larger imbalance, with ~ 75% of thetas being 0, is present.

3.3 Data Representation

In SynthCave, we provide three types of LiDAR data representations of the collected and then processed data: point clouds, images, and graphs. Alongside these representations, IMU and GT data are served as metadata. We synchronize the IMU and LiDAR frequencies by sampling the IMU data down to the LiDAR frequencies through addition. For each representation, the data is partitioned into 14 train sequences and 6 test sequences for each cave section type.

Cave Section Type	Sequences	Duration (s)		XZ-Distance (m)		Y-Distance (m)		Avg. Rotation (°/s)	
		Median	Total	Median	Total	Median	Total	ϕ (horiz.)	θ (vert.)
Default									
Even Path	20	13.80	274.40	23.40	462.42	0.00	0.00	59.42	5.25
Even Path Upwards	20	17.5	351.80	28.84	628.77	17.49	359.12	43.25	10.36
Even Path Downwards	20	16.00	338.20	27.59	578.29	11.0	261.00	22.40	9.84
Advanced									
Entrance	20	13.40	280.60	24.54	521.00	10.25	287.92	27.19	10.53
Curvy Even Path	20	15.40	348.60	26.16	588.01	0.00	0.82	**86.72**	8.24
Curvy Path Upwards	20	16.60	339.00	29.18	604.23	15.39	333.13	60.80	9.65
Curvy Path Downwards	20	16.40	350.60	30.81	628.10	10.00	230.94	57.20	13.20
Miscellaneous									
Underwater	20	**20.10**	**432.8**	**59.84**	**1228.17**	22.22	582.86	61.28	17.41
Mineshaft	20	18.80	402.2	31.86	653.61	1.00	45.64	81.91	8.49
Roping Up Shaft	20	17.00	360.0	10.52	246.84	**43.28**	932.17	69.15	**24.46**
Roping Down Shaft	20	7.50	164.4	9.76	242.89	38.17	**1073.21**	68.47	16.98
Total	**220**	**15.80**	**3642.60**	**27.00**	**6382.31**	**12.59**	**4106.55**	**58.54**	**12.27**

Table 2: Statistics describing the content of SynthCave with the duration in seconds, XY (horizontal) and Y (vertical) distance traveled in meters and ϕ (horizontal) and θ (vertical) as average rotation in degrees per second for every cave section type.

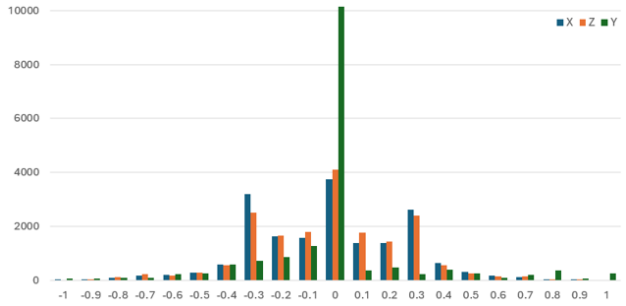

Figure 2: Histogram of the position changes, rounded to 0.1 and limited to 1 and -1, of the GT values of SynthCave. Outside the limit are 3 X, 1 Z and 1035 Y values.

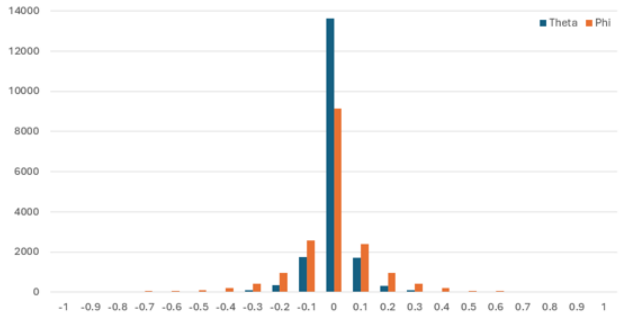

Figure 3: Histogram of the rotation changes, converted to radians, rounded to 0.1 and limited to 1 and -1, of the GT values of SynthCave. Outside the limit are 3 theta and 361 phi values.

3.3.1 Point Cloud. The point cloud representation is being presented in the form of 3D scans, as already formalized in Section 3.1. We normalize the distances by the range of L.

3.3.2 Image. The benchmark's image representation are LiDAR scans transformed into depth images. These images, labeled as $I_{C,L,t}$, hold data from a scan $S_{C,L,t}$. Each depth image is formatted with the resolution of L as an $H_L \times V_L$ grid, similar to standard image resolutions. During the transformation process, the LiDAR-derived distance scans $S_{C,L,t} = (d_{1,1}, d_{1,2}...d_{H_L,V_L})$ are converted into grayscale values ranging from 0 to 1, which generate a visual representation of the spatial data. These depth images $I_{C,L,t} = (g_{1,1}, g_{1,2}...g_{H_L,V_L})$ resemble standard images, but they also contain depth information that corresponds to the distances measured by the LiDAR sensor.

3.3.3 Graph. In the graph representation, we define a cave section C, measured by a LiDAR L, as a temporal graph denoted by $C_L = (G_1...G_T)$. Each instance $G_t = \{N_t, E_t\}$ within this graph represents the scan of L at a given time step $t \in T$, conceptualized as a $H_L \times V_L$ rectangular grid graph, which mimics the resolution of L. Within a grid graph G_t, each node $n_{t,h,v} \in V_t$ corresponds to a distance scan captured by L, where the total number of nodes is determined by $|N_t| = |H_L \times V_L|$. Each node $n_{t,h,v}$ encompasses an attribute vector $a_{v_{t,h,v}}$ that contains the distance normalized by the range of L, the vertical angle and the horizontal angle. A non-border node $n_{t,h,v}$ is connected by four edges from E_t to neighboring nodes $n_{t,h-1,v}$, $n_{t,h+1,v}$, $n_{t,h,v-1}$, and $n_{t,h,v+1}$.

4 BASELINE MODELS

In this Section, we introduce the baseline models used for evaluating SynthCave. The results are provided in Section 6. We consider methods like RANDOM, ZERO and CNNs for baselines, and additionaly, for each data representation type, choose a suitable SOTA deep learning model architecture.

Convolutional Neural Network (CNN) We use the same preprocessing and model architecture as Nicolai et al. [23]. First, we concatenate the depth images $x_1 = I_{C,L,t} \oplus ... \oplus I_{C,L,t-K+1}$ and input the resulting image to the feature extractor. Additionally,

we expand the feature vector f resulting from the feature extractor with the concatenated IMU data $x_2 = IMU_{C,t-K+1} \oplus ... \oplus IMU_{C,t-K+1}$, before forwarding f to the MLP model head. To handle the issue of f becoming too large when increasing K, we replace the last max-pooling layer with adaptive-average-pooling using a shape $(\frac{H}{20}, \frac{W}{20})$, which keeps the parameter count of the MLP head reasonably sized at $|f| = 64 \cdot \frac{H}{20} \cdot \frac{W}{20} + 6 \cdot K$.

Temporo-Spatial Vision Transformer (TSViT) We utilize the *TSViTcls* implementation of TSViT, as presented by Tarasiou et al. [39]. As feature extractor input, we use depth image sequences defined as $x_1 = (I_{C,L,t}, ...I_{C,L,t-K+1})$ with $|x_1| = K$. Afterwards, we merge the IMU data $x_2 = IMU_{C,t} \oplus ... \oplus IMU_{C,t-K+1}$ with the resulting feature vector, similar to the CNN procedure. Additionally, we modify the time embedding method by embedding the frame number instead of the day of the year. We also drop the channels dimension because of gray scale images. Last, we define an input size of 24x24 which is similar to the original paper, and interpolate the image if the input size is different.

Point Spatio-Temporal Network (PSTNet) We use the *NTU* variant of PSTNet, as originally introduced by Fan et al. [6]. To train the model on consumer grade GPUs, we reduce the amount of input points by a factor of 10 by only using every 10th point. We define $x_1 = (S_{C,L,t}, ...S_{C,L,t-K+1})$ with $|x_1| = K$ and handle the IMU data x_2 the same as in the previous models.

Attention Spatial-Temporal Graph Convolutional Network (ASTGCN) We build a custom model based around a feature extractor consisting of 3 ASTGCN blocks, as implemented by Guo et al. [10], with the difference of having a temporal dimension kernel size of 1. The model input is defined as $x_{11} = (V_{C,L,t}, ...V_{C,L,t-K+1})$ and $x_{12} = (E_{C,L,t}, ...E_{C,L,t-K+1})$ with $|x_{11}| = |x_{12}| = K$. Within the model, we first apply TopKPooling [15] with a ratio of 0.05 to the input data. Afterwards, we pass the pooled graph data to the ASTGCN blocks. Next, we reduce the feature dimensions by applying 1D adaptive average pooling. Finally, we merge the IMU data x_2 the same as with TSViT and pass the resulting feature vector into a MLP head.

RANDOM As one of the two lowest bound models, RANDOM does not process any input and produces random outputs that are uniformly distributed in the interval $[-1, 1]$.

ZERO The second lowest bound model, ZERO, also does not process any input and returns only 0 values.

5 EXPERIMENTAL APPARATUS

In this Section, we describe how we apply the baseline models on SynthCave. First, we explain the training procedure in Section 5.1, then our hyperparameter optimization in Section 5.2 and, finally, our evaluation strategy in Section 5.3. It is important to note that the baseline results are based on the use of a single LiDAR sensor, specifically the *Default LiDAR*.

5.1 Procedure

SynthCave's training set is used to create 10 training sequences and 4 evaluation sequences for each cave section type. For the training sequences we generate histograms of the GT distribution similar to Figure 2 and Figure 3. The histograms will be used in Section 5.2 to weight the loss function. We also use the $[-1, 1]$ bounds on the GT values during training.
For the CNN and TSViT models, we use the *Image* representation, PSTNet and ASTGCN use *Point Cloud* and *Graph* respectively.
In a training iteration, also defined as epoch, we apply the models to the training sequences and validate them on the validation sequences. Next to the training loss, we observe the training and validation Mean Squared Error (MSE) in total and also for every element $\in \{posX, posY, posZ, \phi, \theta\}$.

5.2 Hyperparameter Optimization

We differentiate between training, model and data hyperparameters. Training hyperparameters, which we do not want to optimize and rather define statically for simplicity, include 30 epochs, a batch size of 8 and AdamW [20] as optimizer. We define a weighted MSE loss function J with dynamically determined weights based on the predictions. For each prediction element $\in \{posX, posY, posZ, \phi, \theta\}$, the function calculates a weight. This is done by rounding the element to one decimal and mapping it to the training sequence histogram from Section 5.1. We define a single weight for such an element as $1 - (\frac{\text{relative occurrence}}{\text{total occurrence}})$. The weights for all elements are then stacked and used to calculate the weighted MSE loss $J = weights \cdot (prediction - gt)^2$. The initial learning rate (LR) is set to $1e-3$ and adjusted with a learning rate scheduler, which reduces the LR like $new\_LR = 0.1 \times LR$ with $min(LR) = 1e-6$, if the validation MSE does not improve for 5 epochs. Additionally, early stopping is implemented, which can end training before reaching 30 epochs if the validation MSE does not improve for 10 epochs. We omit the tuning of model hyperaparameters, like the head count of TSViT for example, and use the default parameters as presented by the respective model authors. However, we employ a grid-search on the data hyperparameter that defines the input sequence length $K = |x|$ for the three SOTA models with $K \in [2, 4, 8, 16]$.

5.3 Evaluation

For every model that has undergone training, per model type, we choose the one with the best validation accuracy for evaluation. We evaluate them on SynthCave's test set, with respect to the data representation type, and ignore training samples where GT values exceed the $[-1, 1]$ interval. We measure the MSE, the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) of the position and rotation estimations for a better comparison with SOTA models, per cave section type and in total.

6 RESULTS

In this Section, we present the results of the application of the baseline models from Section 4 on SynthCave. Based on the best validation results during training, we use TSViT, PSTNet and ASTGCN with $K = 4$ and CNN with $K = 2$ for evaluation.
In Table 3, an overview of position and rotation MSEs, RMSEs and MAEs is given. CNN and ASTGCN perform the best for rotation estimates, followed by TSViT and PSTNet. As for position estimates, RANDOM is the worst performer, while TSViT and ASTGCN are the best.
Table 4 breaks down the position and rotation estimates into the elements of which it consists and provides their RMSE. Looking at X, Y, and Z, all models perform significantly worse on the height axis Y. This is complemented by the θ estimates, where only the CNN performs slightly better than the ZERO model. In contrast, all models perform better on ϕ than the lowest bound models. Also, just as shown before, the SOTA models have the

Model	MSE		RMSE		MAE	
	POS	ROT	POS	ROT	POS	ROT
RANDOM	0.451	0.354	0.560	0.508	0.560	0.508
ZERO	0.113	0.026	0.229	0.076	0.229	0.076
CNN	0.069	**0.021**	0.172	0.075	0.172	**0.072**
TSViT	**0.063**	0.023	**0.163**	0.075	**0.163**	0.075
PSTNet	0.065	0.024	0.167	0.077	0.167	0.077
ASTGCN	**0.063**	0.023	0.164	**0.074**	0.164	0.074

Table 3: Baseline MSEs, RMSEs and MAEs of position and rotation estimates averaged over all sequence categories.

Model	X	Y	Z	ϕ	θ
RANDOM	0.426	0.508	0.418	0.374	0.334
ZERO	0.087	0.173	0.079	0.045	0.008
CNN	0.054	0.097	0.057	**0.035**	**0.007**
TSViT	0.044	**0.091**	**0.053**	0.038	0.008
PSTNet	0.047	0.092	0.056	0.040	0.008
ASTGCN	**0.043**	0.092	**0.053**	0.037	0.008

Table 4: Baseline RMSEs of all elements, averaged over all sequence categories.

lowest RMSE on position estimates, while the CNN has the lowest RMSE on rotation estimates.

Table 5 contains position and rotation RMSEs itemized into the *Default*, *Advanced* and *Miscellaneous* categories of SynthCave, whose content is described in Table 2. The *Default* category has the lowest rotational change, resulting in the ZERO model having the lowest RMSE. TSViT has the lowest positional error. Category *Advanced* contains more sophisticated movement, resulting in more rotation, which has a clear impact on the models rotation estimations. The ZERO model has a higher rotation RMSE than the SOTA models and the CNN.

Surprisingly, when increasing the sophistication of the movement even more in the *Miscellaneous* category, the gap between both the position and rotation RMSEs of the SOTA models and the CNN towards the ZERO model is becoming smaller again.

Model	Default		Advanced		Misc.	
	POS	ROT	POS	ROT	POS	ROT
RANDOM	0.536	0.501	0.540	0.508	0.598	0.513
ZERO	0.196	**0.042**	0.198	0.072	0.286	0.106
CNN	0.123	0.048	0.131	**0.062**	0.251	**0.098**
TSViT	**0.111**	0.052	0.126	0.065	**0.240**	0.101
PSTNet	0.115	0.051	0.130	0.070	0.243	0.104
ASTGCN	0.114	0.051	**0.123**	0.066	0.242	0.100

Table 5: Baseline RMSEs of position and rotation estimates by sequence category.

7 DISCUSSION

The results show that, in most cases, the SOTA models beat the other models. The SOTA models do not strongly differ in their results, showing that all model types solve the odometry task well. For rotation estimates specifically, CNN seems to perform best. All SOTA models require down scaling of the input data, with TSViT using the least amount of data with an input size of $24x24$. This suggests that the position estimations specifically do not require lots of reference points from LiDAR scans. When looking at rotation estimates, this might be the reason for the CNN performing significantly better in this case, since it does not undergo data downscaling.

Another surprising observation is the selection of K. When training the models on $K \in [2, 4, 8, 16]$, $K = 4$ results in the lowest validation losses for SOTA models. In the case of CNN, it is $K = 2$. This suggests that the relevant data lies within the first 4 frames for the time-sensitive SOTA models. The CNN architecture used in this work does not implement any time-sensitive modules and instead fuses the frames together into a large image. When applying the convolutional layers with a static filter size, this means that per theoretical frame, less features are extracted with K becoming larger. With 2 being the smallest possible K, this suggests that the latest frame is significantly more relevant than the others. A similar argumentation can be used for the SOTA models, but on a slightly larger time scale.

7.1 Key Scientific Insights

An interesting observation during the creation of SynthCave as first 3D odometry estimation benchmark was the low usage of θ. Depending on the sequence types, the vertical Y axis has also not been used as extensively as initially anticipated. Nevertheless, the baseline results indicate that 3D odometry estimation on SynthCave is possible, even with different model architectures. Previous studies on odometry estimation, as presented in related works, have conducted only a limited number of experiments with point cloud and graph-based models. Our work confirms the suitability of these alternative models in this field and places them on par with vision-based models. ASTGCN even beats TSViT and CNN on the RMSE metric of the X and Z axis.

7.2 Threat to Validity

The composition of SynthCave reveals a clear data bias towards a rotational change of 0, particularly for θ, as well as for vertical movement on the Y axis. The extent of this bias varies depending on the cave section type. Such a significant bias can result in distorted outcomes and must be considered when training and evaluating deep learning models. To address this issue, we utilize a dynamically weighted MSE loss during training, which is validated by our evaluation results. Nevertheless, it is apparent that the errors for Y are comparatively higher than those for other elements. When talking about errors, the observation of RMSE and MAE values being very similar suggests that the model errors are uniformly distributed and not many outlier errors that are significantly larger than the rest exist. This observation might even benefit from the data bias towards 0 and the limit of [-1,1]. When examining the baseline models, the SOTA ones provide the most accurate position estimations. Interestingly, the CNN, which is only a well-proven method in this case, is still better for rotation estimation.

7.3 Generalization

SynthCave is a synthetic dataset based on the cubic world of Minecraft. While this may suggest limited generalization outside of this specific world structure, the variety of SynthCaves sequence types fundamentally improves the generalization of the models trained on it, especially when applying SDR. As the first 3D odometry dataset, it is difficult to analyze how well the models generalize to different datasets.

By using SOTA metrics for position and rotation estimates, the results can, however, be compared to other 2D odometry benchmarks. The baseline models, as implemented in this work, can also be applied to different odometry datasets after converting the data accordingly.

7.4 Future Work and Impact

The baseline models we use are as close to the authors implementations as possible. This, however, might not always be suitable, because they were conceptualized for different use cases. In future work, the baseline models performances could be improved by modifications with OE in mind from the beginning. Especially the tuning of the ASTGCN's model hyperparameters offer potential improvements. Similar to the original paper, we use three Chebyshev polynomials, which extract information of the surrounding $(0-2)^{th}$-order neighbors of a node. Depending on the size of different surface regions captured by a LiDAR sensor with a certain resolution, this amount might not be sufficient. Additionally, different models can be tested on the benchmark, potentially beating the current baseline results.

Another angle for improvement could be made within SynthCave itself. The domain randomization process can be improved by applying pyramid consistency across different domains [50], which utilizes auxiliary domains during training to improve model generalization by learning domain-invariant representations. Examples for cave-OE auxiliary domains could be autonomous driving [45] or office [47] scenarios. We also suggest the exploration of active domain randomization [21] with real-world data as testset. Insights from the creation of SynthCave may be used for future 3D odometry datasets. Critical elements to consider for example are that θ is typically very small and Y is often underrepresented.

8 CONCLUSION

The creation of the SynthCave as first 3D odometry benchmark dataset marks a significant advancement in this field. The dataset, which presents LiDAR data in three distinct forms, provides a comprehensive platform for the application and evaluation of various model architectures.

The introduction of baselines for SynthCave, based on well-proven methods like CNNs, and the exploration of state-of-the-art models such as TSViT, ASTGCN, and PSTNet, have yielded promising results. Despite the low usage of θ and the vertical Y axis in certain sequence types, the baseline results indicate the feasibility of 3D odometry estimation on SynthCave. Our work not only confirms the suitability of point cloud and graph-based models in this field but also places them on par with vision-based models.

The implications of this work extend to transforming navigation and mapping in complex, cave-like environments. More precise and efficient 3D odometry estimation could significantly enhance exploration and rescue missions, geological studies, and other applications. The SynthCave dataset could serve as a valuable resource for future research, fostering innovation and advancements in this field.

LIMITATIONS

This section discusses the limitations of our work. While our results may appear promising, it is important to note that our baseline models have not been tested on different datasets or real-world use cases. Additionally, there is a data bias in SynthCave. We address this issue by using a dynamically weighted MSE loss, but the errors for Y are still comparatively higher than those for other elements. Another potential source of bias is the small rotational changes of θ. While these changes may be natural, they do not necessarily improve the situation.

REFERENCES

[1] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1534–1543. https://doi.org/10.1109/CVPR.2016.170

[2] Yukyung Choi, Namil Kim, Soonmin Hwang, Kibaek Park, Jae Shin Yoon, Kyounghwan An, and In So Kweon. 2018. KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving. *IEEE Transactions on Intelligent Transportation Systems* 19, 3 (2018), 934–948. https://doi.org/10.1109/TITS.2018.2791533

[3] James Diebel, Kjell Reutersward, Sebastian Thrun, James Davis, and Rakesh Gupta. 2004. Simultaneous localization and mapping with active stereo vision. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 4. IEEE, 3436–3443.

[4] A. Diosi and L. Kleeman. 2005. Laser scan matching in polar coordinates with application to SLAM. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3317–3322. https://doi.org/10.1109/IROS.2005.1545181

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[6] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. 2021. PSTNet: Point Spatio-Temporal Convolution on Point Cloud Sequences. In *International Conference on Learning Representations*. https://openreview.net/forum?id=O3bqkf_Puys

[7] André O Françani and Marcos ROA Maximo. 2023. Transformer-based model for monocular visual odometry: a video understanding approach. *arXiv preprint arXiv:2305.06121* (2023).

[8] Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* 36, 4 (1980), 193–202.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[12] Robert Huitl, Georg Schroth, Sebastian Hilsenbeck, Florian Schweiger, and Eckehard G. Steinbach. 2012. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. *2012 19th IEEE International Conference on Image Processing* (2012), 1773–1776. https://api.semanticscholar.org/CorpusID:206813006

[13] Shenbagaraj Kannapiran, Nalin Bendapudi, Ming-Yuan Yu, Devarth Parikh, Spring Berman, Ankit Vora, and Gaurav Pandey. 2023. Stereo Visual Odometry with Deep Learning-Based Point and Line Feature Matching using an Attention Graph Neural Network. *arXiv preprint arXiv:2308.01125* (2023).

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]

[15] Boris Knyazev, Graham W Taylor, and Mohamed Amer. 2019. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems* 32 (2019).

[16] KR Koch and Y Yang. 1998. Robust Kalman filter for rank deficient observation models. *Journal of geodesy* 72 (1998), 436–441.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[18] Donghwan Lee, Soohyun Ryu, Suyong Yeon, Yonghan Lee, Deokhwa Kim, Cheolho Han, Yohann Cabon, Philippe Weinzaepfel, Nicolas Guérin, Gabriela Csurka, et al. 2021. Large-scale localization datasets in crowded indoor spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3227–3236.

[19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[20] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[21] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. 2020. Active domain randomization. In *Conference on Robot Learning*. PMLR, 1162–1176.

[22] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. 2010. The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine* 17, 3 (2010), 56–65. https://doi.org/10.1109/MRA.2010.937855

[23] Austin Nicolai, Ryan Skeele, Christopher Eriksen, and Geoffrey A Hollinger. 2016. Deep learning for laser based odometry estimation. In *RSS workshop Limits and Potentials of Deep Learning in Robotics*, Vol. 184. 1.

[24] Alwin Poulose, Odongo Steven Eyobu, and Dong Seog Han. 2019. An indoor position-estimation algorithm using smartphone IMU sensor data. *Ieee Access* 7 (2019), 11165–11177.

[25] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. 2019. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 7249–7255.

[26] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CoRR* abs/1612.00593 (2016). arXiv:1612.00593 http://arxiv.org/abs/1612.00593

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).

[28] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics* 34, 4 (2018), 1004–1020. https://doi.org/10.1109/TRO.2018.2853729

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[30] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. 2016. Learning Transferrable Representations for Unsupervised Domain Adaptation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/b59c67bf196a4758191e42f76670ceba-Paper.pdf

[31] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2016. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. arXiv:1612.07659 [stat.ML]

[32] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12026–12035.

[33] Weijing Shi and Raj Rajkumar. 2020. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1711–1719.

[34] Leyuan Sun, Guanqun Ding, Yue Qiu, Yusuke Yoshiyasu, and Fumio Kanehiro. 2023. TransFusionOdom: Transformer-Based LiDAR-Inertial Fusion Odometry Estimation. *IEEE Sensors Journal* 23, 18 (2023), 22064–22079. https://doi.org/10.1109/JSEN.2023.3302401

[35] Xun Sun, Yuanfan Xie, Pei Luo, and Liang Wang. 2017. A Dataset for Benchmarking Image-Based Localization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5641–5649. https://doi.org/10.1109/CVPR.2017.598

[36] Wennie Tabib, Kshitij Goel, John Yao, Curtis Boirum, and Nathan Michael. 2021. Autonomous cave surveying with an aerial robot. *IEEE Transactions on Robotics* 38, 2 (2021), 1016–1032.

[37] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. 2018. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis. arXiv:1803.10368 [cs.CV]

[38] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. 2017. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications* 9, 1 (2017), 1–11.

[39] Michail Tarasiou, Erik Chavez, and Stefanos Zafeiriou. 2023. ViTs for SITS: Vision Transformers for Satellite Image Time Series. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10418–10428.

[40] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 969–977.

[41] UNIV BRITISH COLUMBIA [CA]. 2004. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. US Patent.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML]

[44] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. 2017. Image-based localization using LSTMs for structured feature correlation. arXiv:1611.07890 [cs.CV]

[45] Xinshuo Weng, Yunze Man, Dazhi Cheng, Jinhyung Park, Matthew O'Toole, and Kris Kitani. 2020. All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. *arXiv* (2020).

[46] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[47] Fei Xia, Amir Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. 2018. Gibson Env: Real-World Perception for Embodied Agents. arXiv:1808.10654 [cs.AI]

[48] Chouchang Yang and Huai-Rong Shao. 2015. WiFi-based indoor positioning. *IEEE Communications Magazine* 53, 3 (2015), 150–157.

[49] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-2018)*. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2018/505

[50] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. 2019. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2100–2110.

[51] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 1234–1241.

[52] Ce Zheng, Yecheng Lyu, Ming Li, and Ziming Zhang. 2020. Lodonet: A deep neural network with 2d keypoint matching for 3d lidar odometry estimation. In *Proceedings of the 28th ACM International Conference on Multimedia*. 2391–2399.

A ABSENCE OF CLASSICAL ALGORITHMS

In the context of this paper, classical algorithms such as ICP or PSM have not been utilized. These algorithms traditionally estimate the transformation matrix from point cloud A to point cloud B. However, a significant challenge arises when attempting to convert this transformation matrix into the 3D coordinate system of the LIDAR, which is responsible for recording the point clouds. The conversion requires a multi-dimensional mapping function that accounts for both the position and orientation of the LIDAR, the intrinsic and extrinsic parameters of the LIDAR system itself, potential errors and uncertainties in the data, and the calculation of the LIDAR's direction vector. These factors combined make the conversion process a non-trivial task that requires careful consideration and implementation. This is why these classical methods were deemed less suitable for the purposes of this study. Instead, the focus of this paper is directed towards deep learning models, with a particular emphasis on graph models. The potential of deep learning in handling complex data structures and patterns is well-documented, and it is believed that these models can provide more robust and accurate results in the context of this research. To ensure the validity and reliability of the deep learning models, they have been compared with both random and zero models. This comparative approach allows for a more comprehensive understanding of the performance and capabilities of the deep learning models.