CAD model reconstruction from the LiDAR scan of the catenary arch

BARTOSZ PRZADKA, University of Twente, The Netherlands

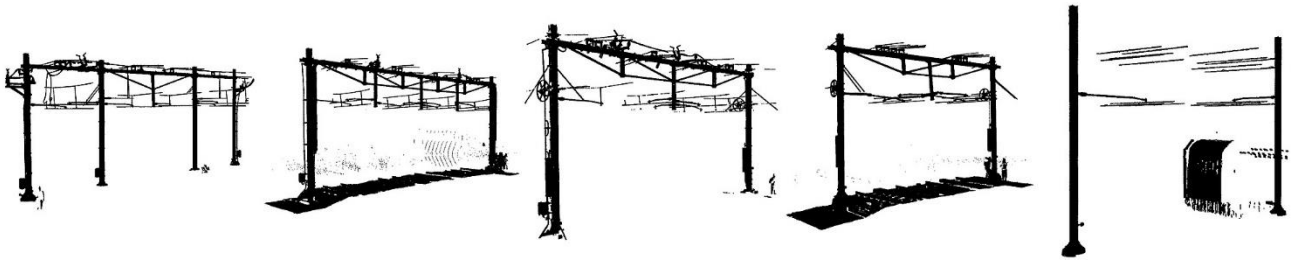


Fig. 1 Exemplary LiDAR-based scans of the catenary arches [1]

The railway industry sees a chance to improve its operations both in terms of security and efficiency by digitalizing the railway tracks and nearby structures. Strukton Rail in collaboration with the University of Twente and the Saxion University of Applied Science began the digitalization of its resources as the necessity for such a process began emerging worldwide. The main focus in this process is the catenary arch as it plays a crucial role in the network's fundamental functionality. Such an arch consists of poles, insulators and many more elements. Currently, the companies managing the network do not have an overview of their infrastructure or the inventory of the elements. Moreover, it can contain both legacy and new elements, which requires trained professionals to perform a manual on-site state assessment and therefore makes the operations cumbersome and time-consuming. This is why there is a need for digital twins (virtual representations) of those, allowing for faster and more reliable repairs and improvements.

This paper explores the process of matching the LiDAR-produced point clouds of the catenary arches to its CAD template stored in the catalogue form. It omits the semantic segmentation in its core as it is perceived as unnecessary and focuses on a deep convolutional neural network (CNN) to extract the features from a scan. Such an approach ensures the labelling task and retraining of the semantic segmentation network are unnecessary. This research is a proof of concept and confirms that the proposed method is feasible and provides numerous improvements over the existing one. To find the best interest point detection method for the given dataset, multiple methods were compared. In the last phase, it was compared to the existing method and provided an overall 250 times speed improvement in terms of adding a new CAD template to the models' catalogue.

Additional Key Words and Phrases: CAD, point cloud, deep learning, catenary arch, feature extraction, CNN, digital twin, railway, reconstruction

TScIT 37, July 8, 2022, Enschede, The Netherlands

1 INTRODUCTION

There is about 3055 km[2] of the railroad network in the Netherlands alone. In this network, there are catenary arches, placed at 50 - 70 m intervals, which give around 43 642 – 61 100 pieces in total. They are fundamental to this system as they carry the power lines, insulators and many more elements above the traction, safely from the users. This study disregards the electric wires and focuses on the construction elements such as poles and arches with the elements assembled upon them.

Those structures are a mixture of both new elements and their legacy equivalents, which makes it difficult for the maintenance team to conduct necessary repairs and improvements in a fast and efficient manner. Currently, the state of the elements and the need for repair are assessed by the specialised workers manually by the means of visual inspection. As the state of the elements is crucial to the safety of the entire network and carried passengers it is essential to explore new methods to map the infrastructure to its digital equivalent (twin). Digital twins are the virtual representations of physical objects with respect to their size, shape and orientation[3]. A new method would allow workers to scan the environment from the train using for instance LiDAR and to find correct elements from its library quickly. This necessity was noticed by Strukton Rail, who in collaboration with the University of Twente and the Saxion University of Applied Science began the digitalization process of the railway network infrastructure.

In the mentioned digitalization phase a new inventory system needs to be proposed. This paper explores the deep learning-based procedure to reconstruct the CAD models, stored in the library form from the point cloud scans of the real-life catenary arches, using deep learning feature extraction. Deep learning is a subtype of the machine learning method. It consists of multiple convolutional layers, each, transforming the input data into a more abstract form, which in the end allows its program to detect objects, features and many more[4]. This study focuses mainly on the point clouds generated by the LiDAR installed on the service train and CAD templates catalogue.

In the following sections, we will provide a problem with the current situation and research questions, supporting the search for solutions. Next, the results of a performed literature review will be provided with relevant information and possible solutions. From the given information the proposed method is derived and will be described in-depth. In the next section, the implementation of the pipeline is provided, all experiments and their results. In the end, we will provide conclusions and information on which aspects of a new method need improvement.

2 PROBLEM STATEMENT

In 2022, there was already performed a study about matching the catenary arch's point cloud-based scan to its CAD templates [5]. However, it requires labelled data in the semantic segmentation part of the pipeline. By using semantic segmentation, when a new element is added to the library it needs to be labelled first and later the network needs to be re-trained, which could cause delays and higher costs for the companies. The proposed improvement is to omit the middle step (semantic segmentation) and to train the deep learning model directly to extract the point cloud's features. Those could be used later to find the corresponding CAD model from the catalogue. Such a change would make the labelling task unnecessary and generate a much simpler and straightforward process.

2.1 Research question

From the above problem the following research questions arise:

- RQ.1 What deep-learning methods are available to extract the point clouds' key features?
- RQ.2 How closest point pairs and corresponding transition matrix can be calculated?
- RQ.3 Is the matching process with no semantic segmentation faster than the previous approach [5] in terms of modifying the template library?

3 RELATED WORK

To find the relevant literature Scopus, Google Scholar, and IEEE Explore sources were used.

Although CAD template matching from the point cloud is a relatively common topic within the academic area, the processing pipeline similar to the proposed one is not. The main difference being most papers use semantic segmentation as a tool to class the labelled points into multiple segments. Such examples are PointNet++[6], introduced in 2017 and RandLA-Net[7] published in 2019. Next, those segments are matched to the sampled point cloud. The most common algorithm to match the template with the cloud is the point pair feature-based matching. Although it works well without template matching involved, it is perceived as a redundant step in the application's context, as mentioned previously.

Other solutions use normal orientation detection or outlier detection in the pre-processing phase and later on a feature description[8]. This could have worked in the scope of the application, however, using deep convolutional neural networks (DCNN) ensures a higher amount of key features found and higher accuracy in the end.

If we consider the feature derivation standalone most algorithms use the density[9] or edge feature extraction[10] to get the representative vectors. Others, less popular ones draw geometrical figures anchoring to the points and generate data from those[11]. According to the performance comparison conducted by Stancelova et al. [12] among the most common detector-descriptor combinations, the most stable detector is Intristric Shape Signatures 3D (ISS3D) and the most performant descriptor is the Fast Point Feature Histogram (FPFH)[13]. Even though those can provide useful matching metrics it provides too little information to match the elements effectively and accurately enough. Therefore, this method is not implemented in this paper.

3.1 Deep feature extraction

In the field of the point cloud, and deep network descriptors very little research can be found. Most methods omit feature extraction standalone and use it inside the neural network itself to segment points [14] for instance.

Others use the network for point cloud registrations[15] or reconstructions [16]. Some of the methods used in the mentioned tasks are relevant to our reconstruction pipeline. One of the examples could be an interest point detection for the point clouds. However, the detection methods used for registration purposes would provide worse performance in our pipeline. The main reason being those networks require multiple fragments of the same point cloud to generate overlap between point neighbourhoods. If it was used with our dataset, the fragments would be required to be generated and therefore the key points detected within those fragments would be random points based on the probability of sampling the same points multiple times. Therefore, the detectors and descriptors with the aim to solve the registration or reconstruction tasks are not adequate in this work.

The most promising work providing both key points detection and description for our point clouds is the Unsupervised Stable Interest Point (USIP)[17]. In the paper, the authors argue their method is the most stable algorithm among random, SIFT-3D[18], ISS[19], Harris-3D[20] and 3DFeatNet[21]. It can be used to register point clouds or classify them, however, the method's purpose is to be used as a standalone network.

All information combined provides an answer to the RQ. 1.

3.2 Mesh reconstruction

In order to reconstruct the CAD model from the scan, one could use a mesh reconstruction instead of template matching. This method uses a point cloud surface to generate geometric figures such as triangles or spheres [22], which combined create a mesh.

In this field, a high amount of research can be found, but most of the papers use a deep neural network to train the model from numerous exemplary meshes and reconstruct the point cloud based on those. Unfortunately, the given output does not provide high accuracy in terms of positioning or surface structure. Hereafter, this method is not a valid proposition for our pipeline.

3.3 CAD template matching

In terms of pipelines for CAD template matching a few works can be found. The most promising is the one introduced by Verga et al. [23]. It starts with a pre-processing module, where outliers are removed and surface normal is estimated. Next, it detects key points and extracts features. Later, it uses the K-Nearest Neighbour (K-NN) algorithm to find pairs of points between the template and scene point cloud. Those pairs are clustered using the DBSCAN

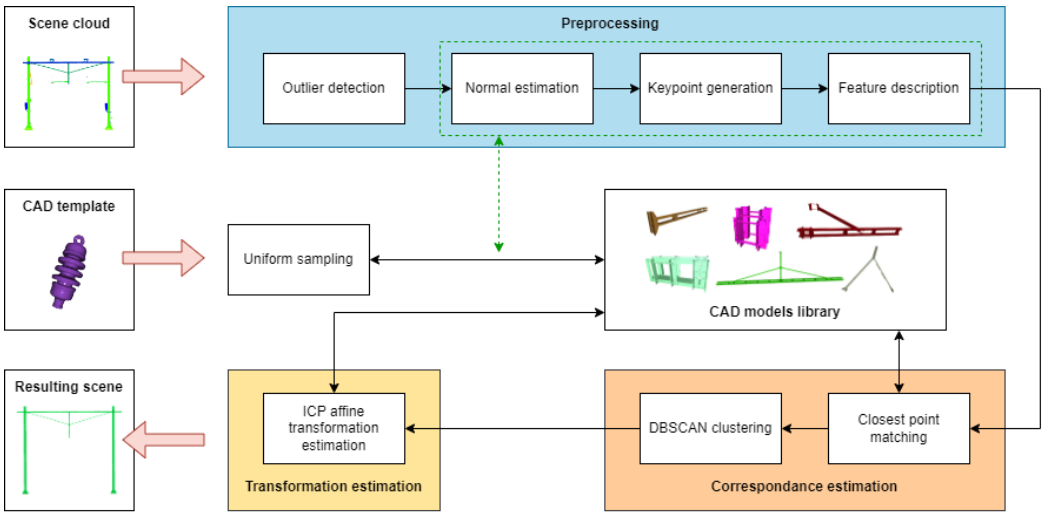


Fig. 2 The overview of the proposed reconstruction pipeline. The green-dotted arrow between uniform sampling and CAD models library indicates that point cloud is prepocessed in the part of the preprocessing modules (green-dotted lines) and later output is added to the library.

method and then it tries to find the minimum-weight matches. In the last stage, the output from the module is used in the Iterative Closest Point (ICP) algorithm to generate the transformation matrix.

The mentioned algorithm provides flexibility in terms of modifying the template library and matching accuracy.

4 DATASET

The data used to provide the results in this study is a dataset, consisting of 15 distinct, high-density terrestrial point cloud scans of the catenary arches (Fig.1) [1]. Those scans were produced using a Trimble TX8 laser scanner placed on the moving maintenance train. It contained single or multiple arches within a single file and all points were manually labelled with one of the 14 classes. It did not contain many outliers as it was previously prepared for the training using the PointNet++[6] network.

In addition to the point clouds, 60 CAD models were used in the models' library. It contained mostly drawings of the arches, poles, insulators, connectors and rods. Examples of used elements are present in Fig. 9. All those elements were distinct, with some having only small discrepancies.

5 PROPOSED METHOD

In this section, we will introduce our method to reconstruct the CAD models. It consists of multiple modules such as preprocessing, correspondence estimation and transformation estimation as depicted in Fig.2.

When referenced to the random transformation of the point cloud its transformations are limited to the rotations and translations in all three axes (x, y, z). The resulting pipeline is inspired by the one proposed by Verga et al. [23].

5.1 CAD template preprocessing

Every model added to the library is processed beforehand. It starts with a uniform random sampling of the mesh with 100 000 or 500 000 points. Next, it goes through the pre-processing pipeline (with the exception of outliers removal) to extract the features, which will be used to match the template with other point clouds. When features are generated, they are saved in files to save the computational time needed to generate those.

5.2 Pre-processing

5.2.1 Outlier detection and normal estimation. To detect the outliers the point cloud is down-sampled using the voxels. Then, the resulting voxels are down sampled again for the specific number of points and outliers are calculated for 15 nearest neighbours and standard deviation equal to 3.

From the processed point cloud surface normal is estimated by querying the k-dimensional tree (KDTree) with a given radius and maximum neighbours number. The default parameters used were a radius of 0.1 and a maximum neighbour number set at 30.

5.2.2 *Keypoint generation.* Two distinct methods are used to generate the interest points. The first one is random uniform points sampling and the second one is the Unsupervised Stable Interest Point (USIP) [17] method.

The first method provides the best performance among the two and when given a high number of input points ensures samples from all scene elements are present and can be matched in later stages. Moreover, it is the only method ensuring the key points can be found in the initial point cloud.

Contrary to random sampling the USIP uses the deep convolutional network to calculate the key points. It provides higher stability and reproducibility of the selected points but does so in a long time. In essence, it takes an input point cloud and creates a randomly transformed copy of it. Next, given points are convoluted with multiple layers, including two of the PointNet [14] layers and normalized to the desired size. Based on the returned points from the normal point cloud and transformed one the Probabilistic Chamfer Loss is calculated to establish the model's performance. The main advantage of this method in comparison to similar solutions [21], [24], [25] is higher point stability and distinctiveness. To find the representative features it performs a

ball query of radius 2, which provides a significant amount of the surface information.

5.2.3 Feature description. After extensive research on the subject of deep convolutional network generating point descriptors, we decided to use the USIP descriptor implementation (Fig.3). This unsupervised deep learning algorithm is similar in its functionality to the 3DFeat-Net [21]. However, according to the paper it outperforms it in terms of speed and flexibility. At its core, it differs from the USIP keypoint detector slightly. The main difference between those in the data input used to train it, is the normal point cloud and its key points, randomly transformed point cloud and its key points and negative samples of the original point cloud. To provide negative samples a random normal point cloud from the data batch (excluding itself by comparing the provided index) is selected as all dataset elements are distinct. Both algorithms use the k-nearest points at their core to find representative features of the points such as density or noise amount. The USIP network uses a ball-search to find the nearest neighbours of the point and based on the output, generates the local descriptor.

In the training phase, random points are selected from the point cloud as it would be computationally impossible to fit the entire point cloud in the GPU's memory with other necessary data. To select those points two methods are used. The first is random uniform sampling and the second is K-th nearest neighbours of a random point. The second method ensures no information is lost from the local environment of the point and density is kept intact. However, for the descriptor to learn efficiently it requires a high number of epochs used in the process. The default size of the selected sample is 15 000 vectors.

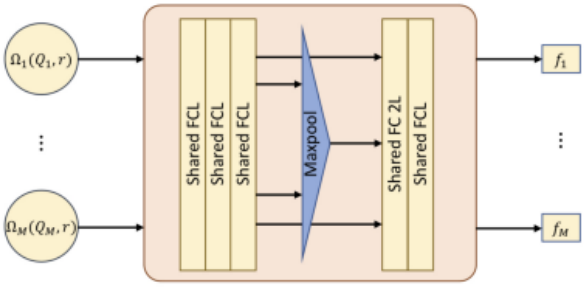


Fig. 3 USIP descriptor network architecture [17].

5.3 Correspondence estimation

5.3.1 Closest point matching. To find the corresponding points between two point clouds the closest point matching module was introduced. It uses the euclidean distance to calculate the similarity between all descriptors derived from the point clouds and returns the lowest value for each point. A sample matching from this module is presented in Fig. 4. The mentioned method provides part of the answer to the RQ. 2.

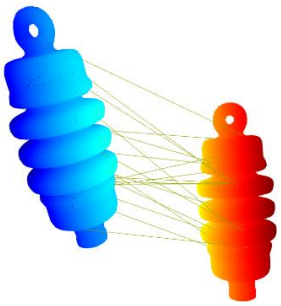


Fig. 4 Example of point clouds registration from the closest point matching. It shows 20 closest pairs based on 128 interest points used.

5.3.2 DBSCAN clustering. The closest point matching function outputs the pairs of closest points between two point clouds. However, those points might belong to multiple instances as their neighbourhoods matches and therefore are classified into the same category. In order to solve this issue, we propose the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm to cluster the points within a single category and therefore perform a simplified instance segmentation. As DBSCAN is the only clustering algorithm not requiring the input number of clusters and is highly performant on diverse input sizes, it is the most suitable candidate among all clustering algorithms [26].

5.4 Transformation estimation

To position the CAD template in the correct orientation, position and scale the Iterative Closest Point (ICP)[27] algorithm is used. In each loop, it calculates the affine transformation matrix between the closest point pairs provided by the correspondence estimation module. The number of loops necessary to position the element depends on the initial distance between point clouds and the initial transformation matrix provided to the algorithm. The mentioned ICP transformation loop provides an answer to part of the RQ. 2.

6 RELATION TO PREVIOUS WORK

The previous work proposed by Vieth [5] focused on the classification and segmentation of catenary arches. In order to add a new template to the catalogue, it needs to be sampled first. Next, the model needs to be retrained in order to learn a new data category.

Contrary to the mentioned pipeline our method's approach to adding new elements is to sample the template, remove outliers, estimate normal, detect key points and generate its descriptors. By using this process no model needs to be retrained, resulting in time savings.

7 IMPLEMENTATION

The implementation of the pipeline was written in the Python language and used multiple external modules. The most important ones were Open3D, Trimesh, Point Cloud Library (PCL) and NumPy. A Trimesh package is a library for working with triangular meshes. Whereas the PCL is an open-source package for point cloud and geometries processing.

Both detector and descriptor use the farthest sampling algorithm to down-sample the input data as the original data size is feasibly impossible to be used directly. This algorithm samples the points selecting ones with the highest distance from the previous one. This ensures better sample coverage over the entire point set.

Fig. 5 Sample scene point cloud with interest points detected by the USIP network.

7.1 Interest point detector

The implementation of the detector, inspired by the USIP paper was developed. It sampled the 10 000 points from the given point clouds using the farthest sampling to train the network. It was trained to return 800 key points by taking into account 32 closest neighbours of the selected points.

7.2 Point descriptor

The original USIP point descriptor expected 3 samples to be provided for training, the source, positive and negative. It was later changed to accept a single point cloud as a data entry and a non-negative sample would be generated by rotating points and translating them randomly. The negative sample was any other point cloud used in the batch. To train it, the minimum batch size used was 4. If the lower size was used, the descriptor could learn a low amount of information as it wouldn't have sufficient negative samples.

7.3 Closest point matching

In the implementation of the closest point matching the K-Nearest Neighbour algorithm is used to find the closest matching point for each point descriptor. In this algorithm, we set a 0.8 distance threshold to ensure higher pair accuracy.

7.4 Transformation estimation

The mentioned ICP algorithm used in the pipeline is the standard version using the maximum iteration number if not converged and the normalization threshold. In our implementation, we set those to 1e-05 and 20 respectively, which ensures lower computational cost overall. Moreover, to speed up the process and reduce the transformation error the initial transformation matrix was provided to the algorithm.

8 EXPERIMENTS AND RESULTS

Both detector and descriptor networks were trained using a remote environment with NVIDIA Quadro P5000 graphic card (16 GB memory), 8 vCPU and 30 GB RAM. Then, for testing purposes the local machine was used with the following specification: Intel i7-11800H, 16GB RAM and NVIDIA GeForce RTX 3050 Ti (4 GB memory) graphic card. The experiments were conducted using Python 3.8, Cuda 11.3 and PyTorch 1.11.

8.1 Detector performance and accuracy

While testing for key point generation with the LiDAR point cloud scene the detector performed poorly, returning points skewed on the left side of the scan (Fig. 5). This behaviour was tried to be corrected by changing the network's parameters and training the network with solely scan point clouds contrary to the default training set. But no action resulted in the correct result.

8.2 Descriptor and closest point matching accuracy

To measure the accuracy of the descriptor and closest point matching algorithm multiple randomly selected CAD models were used with the lowest probabilistic Chamfer loss description model.

The procedure was following, an input CAD model was uniformly sampled with 100 000 points, and then it went through the preprocessing module with a given number of interest points. Next, the same point cloud derived from the model and its interest points were transformed using random rotations, translations and scale, keeping the order of points in the array. The matching interest points from the first point cloud and the transformed ones were saved for later check. Afterwards, for randomly selected interest points within the transformed ones, the descriptors were generated using USIP and inserted into the closest point matching function with the normal point cloud interest points. The closest point pairs (Fig. 4) were compared to the saved corresponding pairs using a ball query of radius 1.5 and later counted. Next, the output number was divided by the total number of the returned point pairs. Those steps were repeated for a different number of interest points (128, 256, 512, 1024) and for each repeated 20 times to increase the accuracy of the results.

The resulting accuracy of the descriptor and pair matching for multiple keypoint sizes were equal to 0.998 with the standard error of 2-04e. This result indicates that the USIP network is a suitable choice for our dataset and ensures state-of-the-art accuracy and stability.

8.3 Descriptor and closest point matching performance

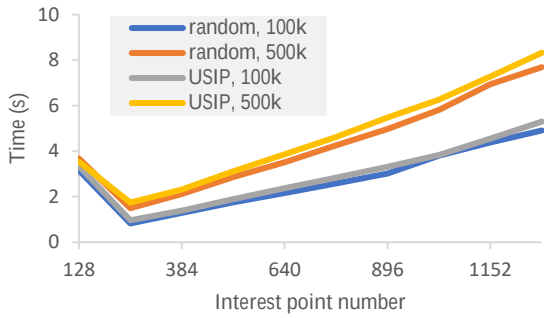


Fig. 6 Measured times to generate interest points and their descriptors in correlation to the interest point number (x-axis), input point cloud size and used algorithm.

In order to measure the performance of the USIP descriptor multiple trials were developed using a different number of interest points, input point cloud size and key point detection method. All configurations were tested over the span of 128 – 1280 interest points with 128 points steps. It was repeated 20 times to achieve higher accuracy of the presented times and different point clouds were used. The used configurations were random key point selection with 100 000 input points and USIP interest point detector with point clouds, having 100 000 and 500 000 points. The detector used in this trial was trained on 15 000 input points and 3000 key points, having an overall loss of 11.5.

From the graph (Fig. 6), the strongly visible linear trend can be derived. For all configurations, a similar tendency follows with different slopes. The slowest description times were noted with the highest input point cloud size and those increase faster with a higher number of key points. An interesting phenomenon can be seen at 128 key points for all settings as its time is equivalent to the one with 640 interest points.

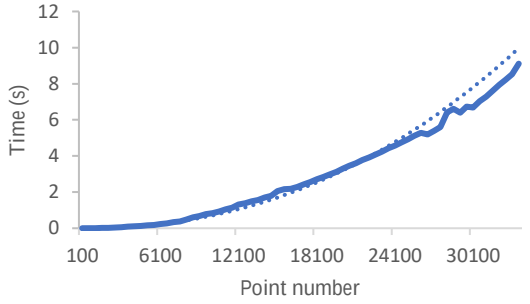
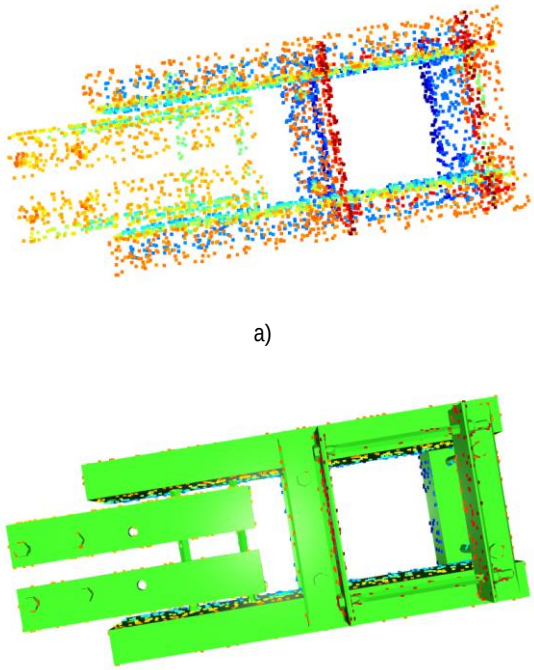


Fig. 7 Measured times to find closest point pairs from two arrays, containing point descriptors. It shows the correlation with the given array size.

With the generated descriptor arrays the performance of the closest point matching was evaluated. All trials were repeated 20 times for higher testing accuracy. With a higher number of points given to the matching module the higher times were noticed. The resulting tendency (Fig. 7) follows an exponential distribution.

8.4 Transformation estimation accuracy



a)

b)

Fig. 8 a) Sample query point cloud. b) Transformed CAD model with source points.

To test the accuracy of the transformation module in our pipeline we followed a similar procedure as the one used in section 8.3. The algorithm used to provide the interest points was a random sampling as it ensures the interest points are present in the pointset. The ICP algorithm provided the transformation matrix, which later on was applied to the source CAD model (Fig.6). The resulting and source point clouds were used to calculate the Root mean square error (RMSE), which was equal to 0.0002. The RMSE reflects the average distance between matching points produced by the correspondence module. It was repeated 20 times and hence the standard error of the 1-05e applies to the provided RMSE.

8.5 New CAD template addition - performance comparison

In order to measure the time necessary to add a new element to the catalogue using the previous method, the model was trained and timed. The average training time for the entire dataset assuming 500 epochs was 1000 seconds overall using the remote environment. While this time is acceptable in normal usage, our template addition takes approximately 2.3 seconds when using 640 random key points and 4s when using 640 USIP key points.

This provides the answer to the RQ. 3 with the positive response. Our approach is approximately 250 times faster than the previous approach. The accuracy of both methods was not compared in the scope of this paper.

9 DISCUSSION AND CONCLUSION

9.1 General conclusion

In this paper, we present a scene reconstruction pipeline for the LiDAR scans of the railway catenary arches. The proposed novelty is semantic segmentation omission in order to simplify the pipeline and make labelling unnecessary. To detect interest points and generate their descriptors it proposes to use the deep neural network, explicitly the USIP model. The pipeline outperforms the existing solution[5] in terms of the time necessary to add a new element into the catalogue and automates most of the tasks necessary to reconstruct the CAD element. To conclude, the chosen method with USIP as the point descriptor, random keypoint selection, ICP transformer and DBSCAN clustering even though having issues, confirms the feasibility to reconstruct the scans of the railway arches.

9.2 Value of this study

The entire study confirmed the hypothesis of semantic segmentation omission. Even though, the results were not as positive as expected they provided insightful data and viability for future work on the subject. Moreover, the resulting pipeline provides the most valuable improvement of all, removing the need for labelling task, which is cumbersome and expensive. Moreover, it showed the performance gains with the proposed solutions over the past pipeline, indicating possible costs and resource savings for the companies.

9.3 Future work

In the future, the pipeline could be improved upon the keypoint detection. To ensure this, the work on higher keypoint stability and reproducibility should be performed. When this is settled it should improve the accuracy of the transformations of the templates as well.

Moreover, work on the own keypoint detector and descriptor could be performed as the research in the field of deep feature extraction is limited and must be explored further.

One other area in need of improvement is the CAD templates overlap in the resulting mesh. As the points belonging to the same element can be classified into multiple elements in the library the elements in the scene might overlap each other. In order to solve this issue, the bounding box of the elements could be used to classify all points within it as the same element.

9.4 Shortcomings and limitations

The biggest issues encountered during the experiment were digital environment issues, mostly with CUDA and PyTorch platforms. Initially, the experiments were expected to be run locally as mentioned in chapter 7, however, the computational power was not sufficient to train the models with lower loss. This resulted in moving the experiments to the cloud environment.

One of the main weak points of this experiment is the environment used to train and test CNN models. With a limited GPU memory, it is not possible to use the entire point cloud as the dataset input and therefore some of the detected key points and generated descriptors might have low accuracy. Additionally, with the lower batch size used in training the models comes the possibility of not having enough distinct negative samples, which can result in high network loss and poor model overall.

Another significant issue encountered in this work was a low number of papers in the field of deep neural network point cloud descriptors. In this field, only a few papers were found and within those only two were suitable to use in the context of this work. As most researchers use feature extraction in their networks directly it is challenging to find a suitable paper.

REFERENCES

[1] B. Ton, "Labelled high resolution point cloud dataset of 15 catenary arches in the Netherlands." 4TU.ResearchData, 2022. [Online]. Available: https://data.4tu.nl/articles/dataset/Labelled_high_resolution_point_cloud_dataset_of_15_catenary_arches_in_the_Netherlands/17048816/1

[2] E. Commission and D.-G. for M. and Transport, *EU transport in figures : statistical pocketbook 2021*. Publications Office, 2021. doi: doi/10.2832/27610.

[3] M. Jacoby and T. Usländer, "Digital Twin and Internet of Things—Current Standards Landscape," *Applied Sciences*, vol. 10, no. 18, 2020, doi: 10.3390/app10186519.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[5] Z. J. Vieth, "Point cloud classification and segmentation of catenary systems." Feb. 18, 2022. [Online]. Available: http://essay.utwente.nl/89565/

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," 2017, doi: 10.48550/ARXIV.1706.02413.

[7] Q. Hu *et al.*, "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds," 2019, doi: 10.48550/ARXIV.1911.11236.

[8] D. Varga, J. M. Szalai-Gindl, B. Formanek, P. Vaderna, L. Dobos, and S. Laki, "Template Matching for 3D Objects in Large Point Clouds Using DBMS," *IEEE Access*, vol. 9, pp. 76894–76907, 2021, doi: 10.1109/ACCESS.2021.3082848.

[9] L. Wang and B. Yuan, "Curvature and density based feature point detection for point cloud data," in *IET 3rd International Conference on Wireless, Mobile and Multimedia Networks (ICWMNN 2010)*, 2010, pp. 377–380. doi: 10.1049/cp.2010.0694.

[10] J. Sun, Y. Jiang, J. Jiang, and X. Bai, "Triangular mesh construction based on point cloud matrix and edge feature extraction," in *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, 2016, pp. 275–279. doi: 10.1109/ITNEC.2016.7560364.

[11] J. Huang and C.-H. Menq, "Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 268–279, 2001, doi: 10.1109/70.938384.

[12] P. Stancelova, E. Sikudova, and Z. Cernekova, "Performance Evaluation of Selected 3D Keypoint Detector–Descriptor Combinations," 2020, pp. 188–200. doi: 10.1007/978-3-030-59006-2_17.

[13] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217. doi: 10.1109/ROBOT.2009.5152473.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." arXiv, 2016. doi: 10.48550/ARXIV.1612.00593.

[15] S. Ao, Q. Hu, B. Yang, A. Markham, and Y. Guo, "SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration." arXiv, 2020. doi: 10.48550/ARXIV.2011.12149.

[16] Y. Li, A. Dai, L. Guibas, and M. Nießner, "Database-Assisted Object Retrieval for Real-Time 3D Reconstruction," *Computer Graphics Forum*, vol. 34, no. 2, pp. 435–446, May 2015, doi: https://doi.org/10.1111/cgf.12573.

[17] J. Li and G. H. Lee, "USIP: Unsupervised Stable Interest Point Detection from 3D Point Clouds." arXiv, 2019. doi: 10.48550/ARXIV.1904.00229.

[18] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM international conference on Multimedia*, 2007, pp. 357–360.

[19] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3D object recognition," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009, pp. 689–696. doi: 10.1109/ICCVW.2009.5457637.

[20] I. Sipiran and B. Bustos, "Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes," *The Visual Computer*, vol. 27, pp. 963–976, Nov. 2011, doi: 10.1007/s00371-011-0610-y.

[21] Z. J. Yew and G. H. Lee, "3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration," in *Computer Vision – ECCV 2018*, Springer International Publishing, 2018, pp. 630–646. doi: 10.1007/978-3-030-01267-0_37.

[22] I. Lang, A. Manor, and S. Avidan, *SampleNet: Differentiable Point Cloud Sampling*. 2020. doi: 10.1109/CVPR42600.2020.00760.

[23] D. Varga, J. Szalai-Gindl, B. Formanek, P. Vaderna, L. Dobos, and S. Laki, "Template Matching for 3D Objects in Large Point Clouds Using DBMS," *IEEE Access*, vol. PP, p. 1, May 2021, doi: 10.1109/ACCESS.2021.3082848.

[24] C. Choy, J. Park, and V. Koltun, "Fully Convolutional Geometric Features," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8957–8965. doi: 10.1109/ICCV.2019.00905.

[25] D. Streiff, L. Bernreiter, F. Tschopp, M. Fehr, and R. Siegwart, "3D3L: Deep Learned 3D Keypoint Detection and Description for LiDARs." arXiv, 2021. doi: 10.48550/ARXIV.2103.13808.

[26] S. Chakraborty, N. K. Nagwani, and L. Dey, "Performance Comparison of Incremental K-means and Incremental DBSCAN Algorithms." arXiv, 2014. doi: 10.48550/ARXIV.1406.4751.

[27] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The Trimmed Iterative Closest Point algorithm," in *2002 International Conference on Pattern Recognition*, 2002, vol. 3, pp. 545–548 vol.3. doi: 10.1109/ICPR.2002.1047997.

[28]

A SAMPLE CAD ELEMENTS

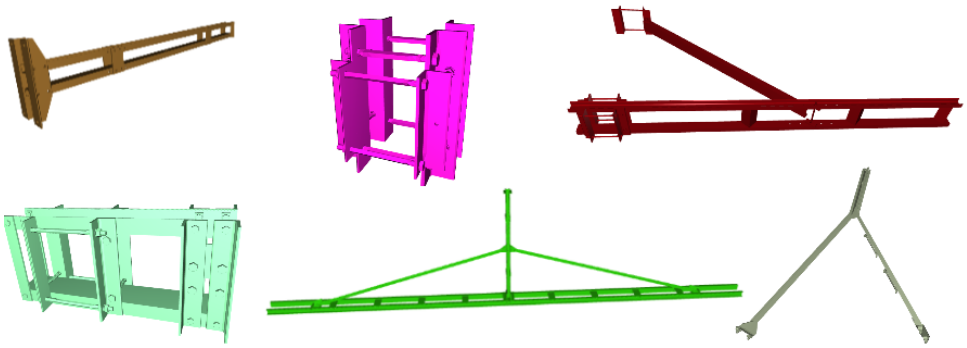


Fig. 9 Sample overview of the CAD elements in the dataset.