Deep Neural Network Robotic Mapping based on 3D Scan

Ong Boon Ping, Cao Liang

Institute of System Science

*Abstract*—Currently, SLAM can be performed using various method such as Kalman filter, particle filter and K-Means method. With development of deep neural network, it is possible to train a neural network that can predict the mapping based on point cloud or lidar data. By using the convolutional neural network, it is possible to mimic the features of particle filter and K-Means method. By training the neural network against the scan data on known terrain or obstacle, the neural network can make prediction based on the 3D scan data. After the training, deep neural network can be deployed on autonomous land or sea vehicle to predict the surrounding terrain. Hence, we propose a deep neural network for the robotic mapping on 3D point cloud scan point data. This deep neural network is generating the prediction on unknown grid points which are not in proximity of any scan point. The neural network is detecting object in different complex shapes. After the prediction, the visualization can be made by displaying the 2D map of the explored region on GUI. Front and rear view can also be generated and displayed on the GUI of the interface as well.

*Keywords*—SLAM, K-Means

# INTRODUCTION

S

IMULTENEOUS Localization and Mapping (SLAM) for an autonomous robot moving in an unknown environment is to build a map of this environment while simultaneously using this map to compute its location [1].

Point cloud data and Lidar data are mostly used in SLAM. Point cloud can be generated by either LADAR (Laser Detection and Ranging) or photogrammetry using camera. [2] The data is returned in Cartesian coordinate with respect to the scanner or depth and angle with respect to the scanner.

Lidar [3] is also used to detect the depth of an object at the angle with the respect of the scanner position. By translating light reflection time from an object into distance, Lidar can measure depth of an object. When the Lidar is capable to detect light reflected at multiple angle, it can measure distance of multiple object point almost simultaneously with the respect of Lidar position. The lidar data can be returned in Cartesian coordinate with respect to the scanner or depth and angle with respect to the scanner.

Point cloud and Lidar are subjected to error due to weather. At the same time, position of point cloud scanner and Lidar is determined by GPS or other positioning system while IMU or gyrometer is used to determine the orientation of point cloud scanner and lidar. GPS and IMU are subjected to error. This complicates the mapping of the surrounding terrain.

K-Means clustering method is used on laser range sensor data for robot mapping [4]. Kalman filter and particle filter are also used in existing simulation or real time application.

With the development of deep neural network, neural network can be created to predict the location of obstacle and map the terrain within bound of scan points.

Convolutional neural network [5] is suitable for mapping problem. This is because convolution on the nearby grid points will be a good approximation of relationship in between the grid points.

In Keras, MaxPooling2D layer can be defined with any pooling size. Consider a scan point is surrounded by unknown grid points, MaxPooling2D allows neighboring unknown grid points to carry the scan point value and proceed with other convolution. Since MaxPooling2D has been largely used to identify the features in images [6], it is possible to use MaxPooling2D layer during the mapping.

Keras also provides Upsampling2D layer to quickly pad the input 2D array and output the upsized 2D array. This can be used in pair of MaxPooling2D layer. This resolve the issue where output is downsized after getting through the MaxPooling2D layer.

Conv2D layer [7] can provide convolution on its input. With more neurons on a Conv2D layer, there will be multiple outputs from the Conv2D layer. Each representing different prediction of the final mapping. With different kernel values, it approximates the predictor function of a terrain based on the input scan points.

Hence, it is possible to convert the scan points into a 2D array and fit the 2D array into neural network for fast computation of mapping.

# Related Work

Several approaches are available for SLAM. K-Means clustering [2], particle filter [8], 6D SLAM [9], Kalman filter [10-11] and GraphSLAM [12] are the common SLAM approaches.

## Clustering based methods

K-Means clustering method defines K number of clusters [13] and make prediction based on cluster boundary. It is able to define the cluster boundary and make prediction based on the cluster center point and spreading. However, a fixed K value is not working for all the terrain. Efficiency of K-Means clustering varies with the K-value [14]. With higher K-value, accuracy can increase but the computation power is also increasing.

There are other cluster methods such as ClusterSLAM [15] which is using distance of points to source as matrix. Hierarchical agglomerative clustering is then applied on the scan points and define the cluster area. However, hierarchical agglomerative clustering may be very sensitive of good initialization [16] and requires long computation time [17].

Hence, clustering can be good in defining clusters of neighboring points with similar height. However, defining suitable cluster number and calculating distance can consume a lot of processing time.

## Kalman filter

The SLAM based on EKF (Extended Kalman Filter) is mainly based on the recursive Bayesian state estimation theory. The extended Kalman filter is a generalization of Kalman filter in nonlinear systems [21].

The SLAM algorithms based on Kalman filter can be summarized as an estimation of iterative correction process [22]: firstly, predict the position of the robot at the present moment by the robot's position at the previous time, and predict the environmental feature of observation, then, calculate the difference between the actual observation and the estimated observation, calculate Kalman gain K with variance, and correct the robot's predicted position by K, finally, add the new observed environmental features to the map.

Extended Kalman Filter has high convergence and handle uncertainty in the terrain well. [23]

In the standard extended Kalman Filter for SLAM, linearization errors produce inconsistency problems [24]. Kalman Filter is assuming the noise model to be Gaussian noise and hence not applicable to all the situation. [23]

## Particle filter

Particle filter [3] starts from N independent landmark estimators with a set of M particles. There are four steps to recursively updating the particle filter given a new control and observation. Firstly, sample a new robot path given the new control. Secondly, update landmark filters corresponding to the new observation. Thirdly, assign a weight to each of the particles. Lastly, resample the particles according to their weights [18].

The advantage of particle filter method is to handle non-Gaussian noise and non-linearilities. [25-26]

FastSLAM algorithm [19] is using the particle filter method. It is a popular method in estimating the location of obstacles.

The disadvantage of FastSLAM is its inability to maintain particle diversity over long periods of time [20].

## Particle filter and EKF Combination

Particle filter handles non-linearity and non-Gaussian noise well while Kalman filter has high convergence. Hence, it makes it possible to combine both methods together. [27]

Grid Mapping or Gmapping is the notable example [28] of using Rao-Blackwellized Particle Filters with Gaussian assumption at each particle. Since there are multiple particles at one time, it is still able to handle non-Gaussian noise.

Gmapping is used in ROS and a popular method. However, not all the system is using ROS for the mapping.

Complex shaped object may not be well handled by Gmapping. [29]

## GraphSLAM

GraphSLAM is a Simultaneous localization and mapping algorithm which uses sparse information matrices produced by generating a factor graph of observation interdependencies [30].

Since GraphSLAM is landmark based and consider robot pose in building the graph, it requires more computation time or resource than other methods. [31]

## Neural Network

Since convolutional neural network can perform convolution of neighboring spatial inputs, it approximates different noise models at different particles as in the case of particle filter.

The nearby grid points can also go through MaxPooling2D layers and Conv2D layers to achieve effect that is similar to clustering around multiple scan points. [32]

Deep neural network is also free from linearity problem since the result of several conv2D layers can be concatenated.

Hence, by reviewing the related work, deep neural network can be used to perform the mapping task based on the 3D scan data.

# Neural Network

## Idea

In this project, we are converting scan points into 2D Numpy array before input to the convolutional network. The output will be in the form of Numpy 2D array that represents the predicted mapping.

With the scan points, prediction can be made on unexplored grid point through the convolutional network.

## Dataset and Preprocessing

Based on the idea, scan data of 640 walled surfaces are gathered. Actual 2D maps serves as expected output while point cloud scan point maps as input during training.

In this neural network, pre-processing is required. This is to covert the grid points into 2D array with interpolations in between scan points and the sensor position.

The interpolation is limited to the grid point which is near to the ground level. Interpolations are not allowed for scan points with height greater than 20 pixels. This prevents wrong prediction of ground points as elevated points. It will remain as explored terrain.

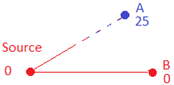


Fig. 1. Interpolation in between scan point to source. Interpolation is not made on points with height beyond 20 pixels since it is more likely to be a space rather than an obstacle. Full interpolation is made on ground or low-lying points during pre-processing.

## Design and Network Architecture

The design architecture consists of 2 phases. The first phase is interpolation area identifier and the following stage is the interpolation value refiner.

Fig. 2. Overall Design Architecture.

Interpolation area identifier focus on identification of unknown points that are in the interpolation bound. A point which is out of interpolation bound can be wrongly predicted based on nearest scan point value.

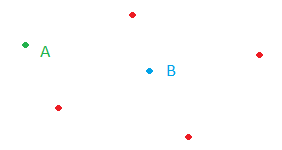


Fig. 3. Interpolation area identifier. Unknown point B is within the interpolation bound since the value can be predicted. Unknown point A is not predicted in this system since it is lack of interpolation information and not observable from source point.

To prevent the error, the neural network must have layers of MaxPooling2D and Conv2D filter layers with larger stride. MaxPooling2D layer allows maximum value in the 64x64 pixel area to be filled in with maximum values. Conv2D layer will generate different kernels at each neuron and hence create different possible 2D maps at each Conv2D layer.

In order to fasten the process of approximation of interpolated area, MaxPooling2D layer is used before Conv2D layer. It allows unknown area to be filled with maximum value and the passing thru Conv2D layer for fine tuning. Area with no scan point will not be predicted.

All the Conv2D layers are using ReLu activation. It allows faster computation and faster learning rate than other non-linear activation functions.

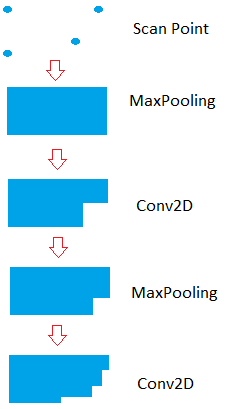


Fig. 4. Approximation of interpolated area after going through Max-Pooling and Conv2D filters twice

In this case, stage 1 is using 64x64 stride for the MaxPooling2D and Conv2D filter layers. After the training, the neural network can learn the trend and discount the pixels which are out of interpolation bound.

At stage 2, input is passing through 32x32 MaxPooling2D and Conv2D filter layers. On the other hand, output of stage 1 is upsized by 2x2 and concatenate with the output of stage 2.

At stage 3 to stage 5, it is using the same structure in stage 2. Hence, 5 stages of MaxPooling2D and Conv2D filter layers are used for the interpolation area identifier phase. Space with unknown value in the filter covered areas will obtain the predicted value closed to the nearest known value after going through the 1st phase of neural network.

Strides at each stage reduced so that the covering area is getting smaller at each stage. Spatial resolution of the interpolated area will become more accurate step by step. As a result, stride 64x64, 32x32,16x16,8x8,4x4 are used for stage 1 to stage 5.

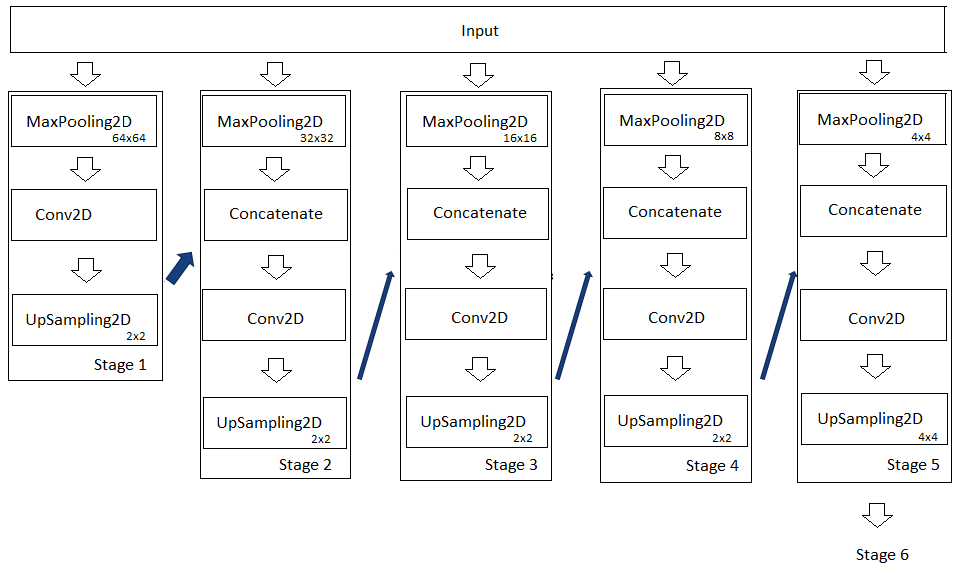


Fig. 5. Neural network blocks from Stage 1 to Stage 5

Output of stage 5 is upsized by 4x4 and concatenate with the input. This concatenated array will be input into interpolation value refiner (Green block in Fig. 8).

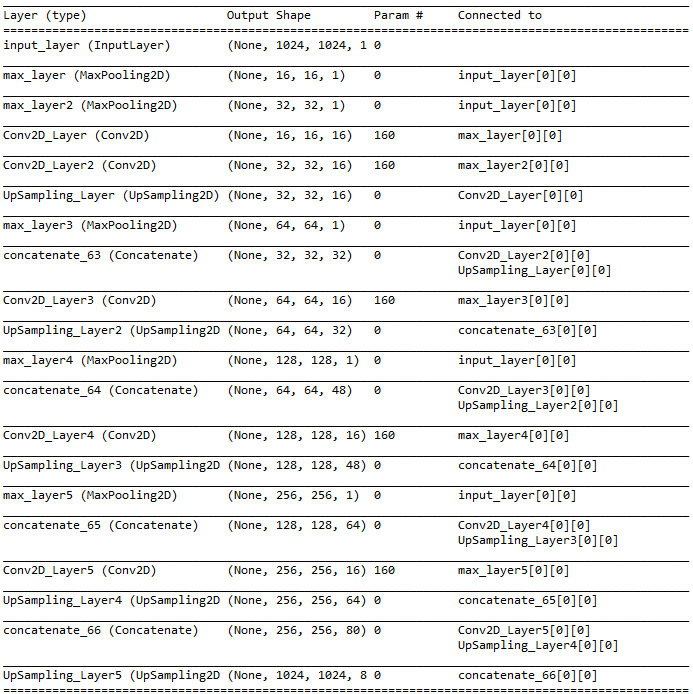


Fig. 6. Neural network layers from Stage 1 to Stage 5

Interpolation value refiner is used here to refine the predicted value. This is because MaxPooling2D network is giving maximum value for the neighboring pixels. The predicted value can be higher than the expected value.

Hence, at stage 6, MaxPooling2D filter is no longer be used at the final layers of the neural network. Conv2D are used in the final layers to smoothen the values across the neighboring grid points using strides 3x3. With small stride value, the points which is outside the interpolation area, will not be getting a prediction value at the final phase due to smooth action.

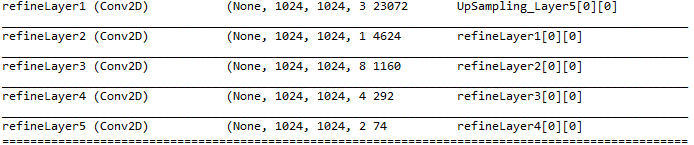


Fig. 7. Neural network layers from Stage 6

After few rounds of training, the design is finalized. The neural network accepts 2D array that contains scan point as input. To preserve the input scan point value, lambda layers are added so to replace the known point pixel value with the measured scan point data.

The final output will give final value or each grid point. To reduce the processing time, the map will be saved in the form of PNG images.

Upsampling

Upsampling

Upsampling

Upsampling

Upsampling

Fig. 8. Neural network Structure.

The design of stage 6 is further enhanced by changing the Conv2D with 6 residual blocks. Each residual block is having 2 Conv2D layers with 3x3 kernel. With this enhancement, the accuracy is increased because deeper network provides better approximation to the real terrain. Residual blocks also reduce overfitting and hence provide better performance than simple Conv2D layer [3].

In this paper, we called this as RES layer enhanced network since stage 6 is using residual layer.

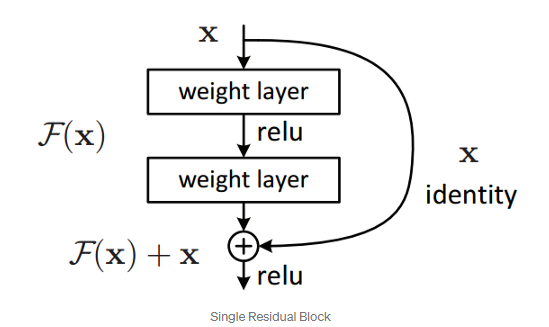
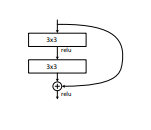
 

Fig. 9. Single residual block [3]. 6 residual blocks are used in the RES layer enhanced neural network.

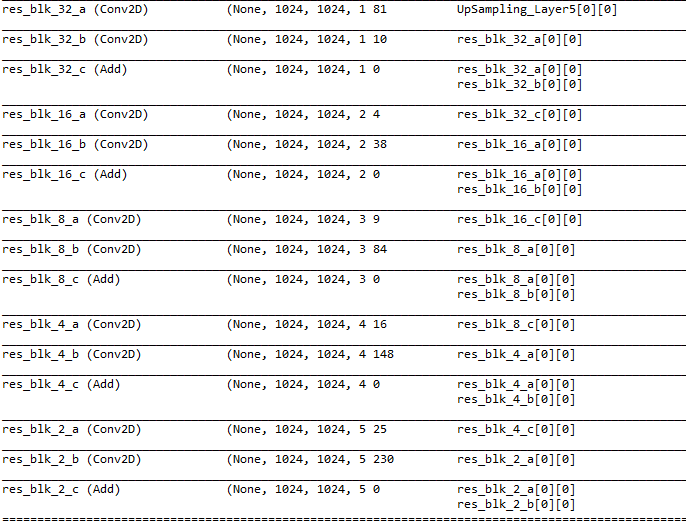


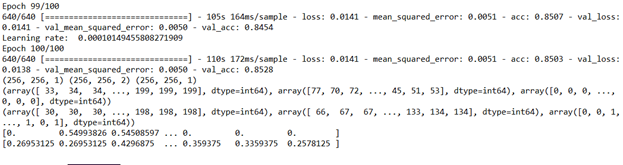
Fig. 10. Residual layers as improvement to stage 6 neural network. (Replace layers shown in Fig. 7)

# Network Training Result

The neural networks are trained using 640 2D maps generated from Lidar or Point Cloud data. Each map is taken from lidar or point cloud scan on wall and various building surfaces. The scan points are all collected from a single point cloud scan. The sensor is placed at different angle from the wall or building surfaces.

Each 2D map has a matching terrain map. As a result, the 2Dmap serves as input to the network while the expected output is the matching 2D terrain map.

After 100 epochs, the accuracy reaching 85% with mean squared error reduced to 0.0051. In this training, we are using RMSprop as Keras optimizer and root mean square as error metric.



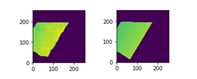


Fig. 11. Non-RESNET Network Training Result. Output map at the left and expected output map at the right.

Similar training is also imposed on RES layer enhanced neural network. After 100 epochs, the accuracy reaching 86.2% with mean squared error reduced to 0.0044. In the training, we are using RMSprop as Keras optimizer and root mean square error as error metric. The RESNET enhanced neural network is having a better result than non-RESNET enhanced neural network.

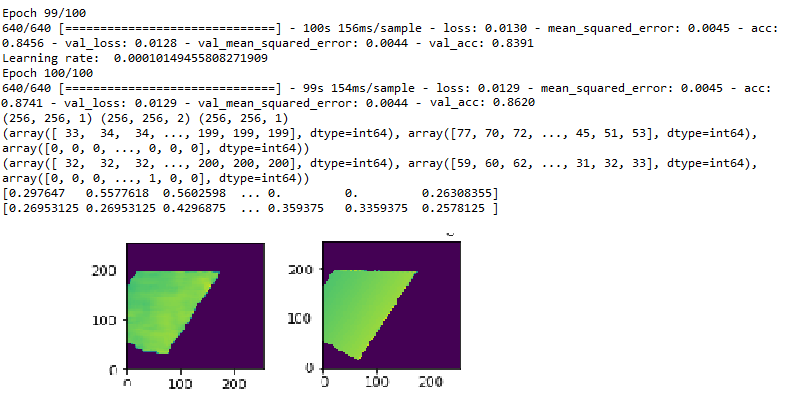


Fig. 12. RESNET Enhanced Network Training Result. Output map at the left and expected output map at the right.

The neural network weight and model are saved in hdf5 format. This will enable the network to be called during the testing stage.

# Network Testing Result

The neural network is applied on the point cloud scan data gathered in Leibnitz University using autonomous robot. [4]

During testing, the network model and weights are preloaded during system initialization. Python script will be converting point cloud scan data into 2D map.

In this dataset, Euler angle and GPS position are appended along with all the scan point in local Cartesian coordinates. Hence, conversion is made using the rotation matrix. [33]

When light or waves can still be reflected from a scan point to the sensor through the space under objects such as tree leaves, these spaces are usually maneuverable. To prevent wrong prediction of space as object at high ground, we are inverting the height values according to Fig. 13 before entering neural network. This prevent MaxPooling2D to be more sensitive with the points with higher value.

The prediction output can be converted back to original height value system later.

During testing, maximum height is clipped to 191. At inverted scale, values below 64 are allocated to be unknown values. The ground will be converted to 255 in value.

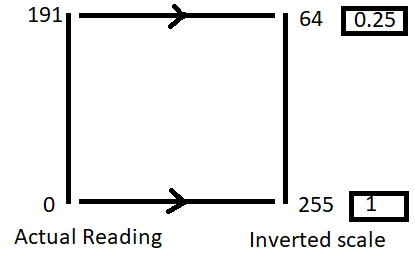


Fig. 13 Conversion of height value before input into neural network. Nominal value is boxed.

Interpolation is made on the newly created 2D map from the scan point to the source point.

After data conversion, preloaded neural network will be used to predict the terrain. Due to limited computing power and time limitation, input map will be processed in 1024x1024 pixel region. Before accessing the neural network, the values are nominalized as shown in Fig. 13. Output map will be saved as PNG image.

After all the regions are processed by neural network, all the PNG image will be combined into one single output terrain map.

Since there are multiple scan files, previous output saved in PNG image will be preloaded before subsequent scan points are loaded. The latest scan points will be overriding the existing values.

Fig. 14. Network Testing Process.

The neural network predicts the height value in unknown grid points. The testing accuracy is 93% when comparing the generated map against the expect mapping.



Fig. 15. Expected mapping.

Inaccuracy comes from noise reflected by the tree leaves and other objects. Due to slicing, certain bordering pixel is not showing intended pixel.

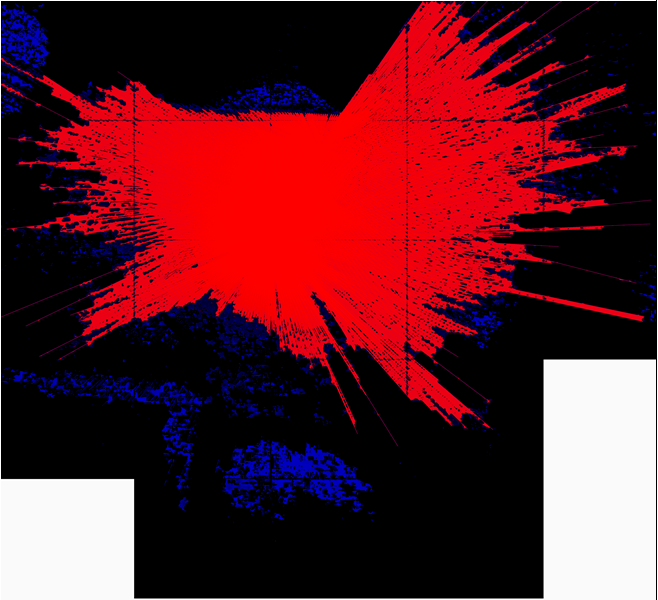


Fig. 16. Neural network generated map without edge noise reduction.

The error at the bordering pixel at the edge of 1024x1024 error can be resolved by padding. After padding all the edges with nearest values, the input and output shape become 1088x1088 pixels.

After each prediction, the padded edges are removed at the output. Hence, the output of 1024x1024 pixels will be returned. It will be free from noise at the edge of 1024x1024 boundary.

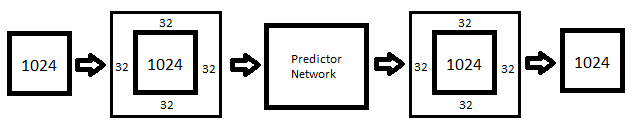


Fig. 17. Edge noise reduction

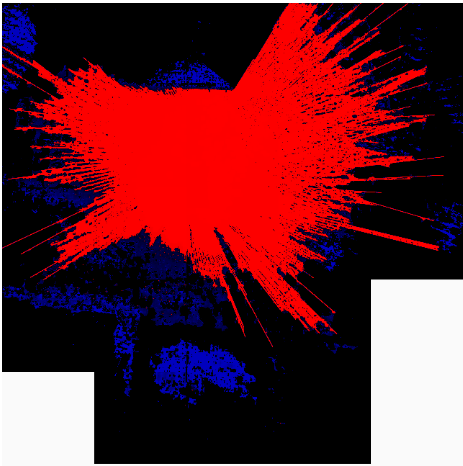


Fig. 18. Neural network generated map with edge noise reduction.

The RESNET refined neural network has a higher testing accuracy of 94.8% when comparing the generated map against the expect mapping.

Noise reflected by the tree leaves and other objects is reduced in the map generated by RESNET refined neural network. It is also more sensitive to border noise, but it can be cleared by using padding on the input.

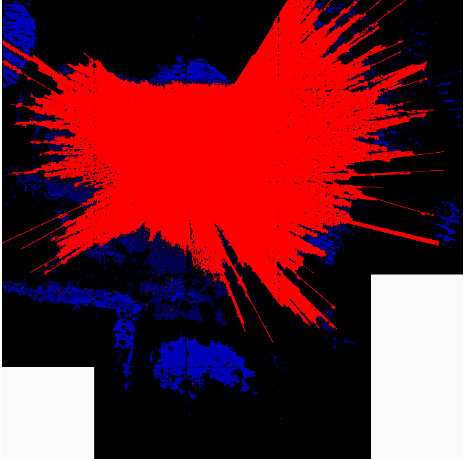


Fig. 19. RESNET enhanced neural network generated map without edge noise reduction.

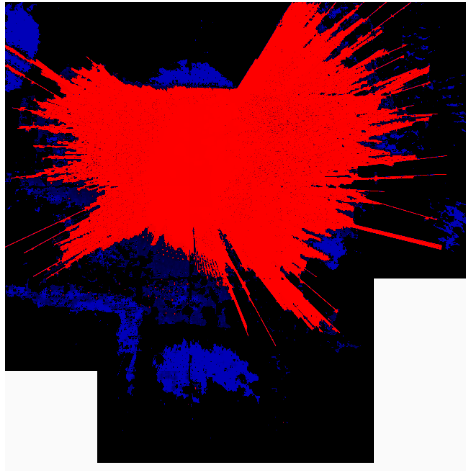


Fig. 20 RESNET enhanced neural network generated map with edge noise reduction.

K-means algorithm [2] is also used to generate the result. The testing accuracy is 70%. The method is considering the 7 clusters or K=7.

Hence, neural network can generate the 2D mapping better than the K-means filter on Lidar/Point cloud scan data.

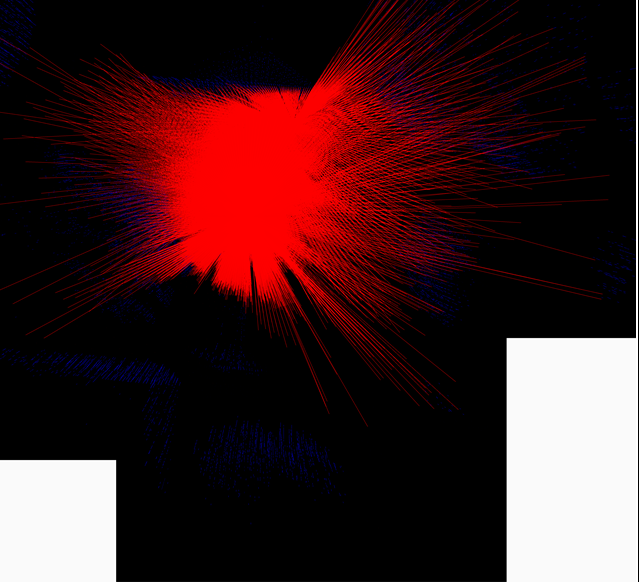


Fig. 21. K-means generated map.

# Result And Discussion

|  |  |  |
| --- | --- | --- |
|  | Accuracy | RMS Error |
| Neural Network (without RESNET) | 85.0% | 0.0051 |
| RESNET Enhanced Neural Network | 86.2% | 0.0044 |

Table 1. Network Training Result. The RESNET enhanced neural network is having a better result than non-RESNET enhanced neural network

The RESNET enhanced neural network is giving a better training result than the non-RESNET enhanced neural network. This is because residual layers manage to reduce the degradation during the training. It is also gaining more accuracy with deeper layers.

|  |  |
| --- | --- |
|  | Accuracy |
| Neural Network (without RESNET) | 93.0% |
| RESNET Enhanced Neural Network | 94.8% |
| K-Means | 70% |

Table 2. Testing Result. The RESNET enhanced neural network is having a better result than non-RESNET enhanced neural network

In this case, the K-Means method is using 7 clusters. Since distance in between scan points increases with its distance to the point cloud scanner, K-Means method is not able to make prediction accurately when the distance between scan points are further away and will be categorized to other clusters.

In comparison, neural network is pre-trained against the real obstacle and terrain. Moreover, 1st MaxPooling2D layers have a pooling area of 64x64 pixel. It allows the prediction to be made in between scan points that have distance of 64 pixels.

Hence, neural network is performing better than the K-Means method.

The RESNET enhanced neural network is performed better since it has deeper layers than the non-RESNET enhanced neural network.

# Visualization

## System Design

The prediction output of the Keras neural network is visualized using Open3D library and Tkinter python packages.

The Map Tracking Viewer GUI sends robot position and orientation data to the backend services through Rpyc remote procedure call. The Open3D Visualizer generates the Map/Front/Rear view based on robot position and orientation, and subsequently sends back the Map/Font/Rear view to the backend service. The Map Tracking Viewer finally receives the Map/Front/Rear view and display them on the GUI.



Fig. 22. Overall Architecture.

## Open3D Visualizer

The Open3D Visualizer creates the 3D mesh based on the grayscale 2D map generated by the Keras neural network. The 2D map uses grayscale value to represent the height of each point.

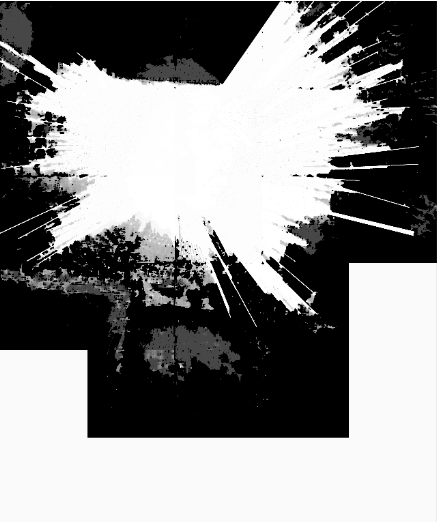


Fig. 23. Grayscale 2D Map.

The Open3D Visualizer [40] reads the robot position and orientation from the backend services, and generates the 2D Map, Front / Rear view pictures dynamically using Open3D library. Finally, all the generated view pictures are sent back to the backend services.

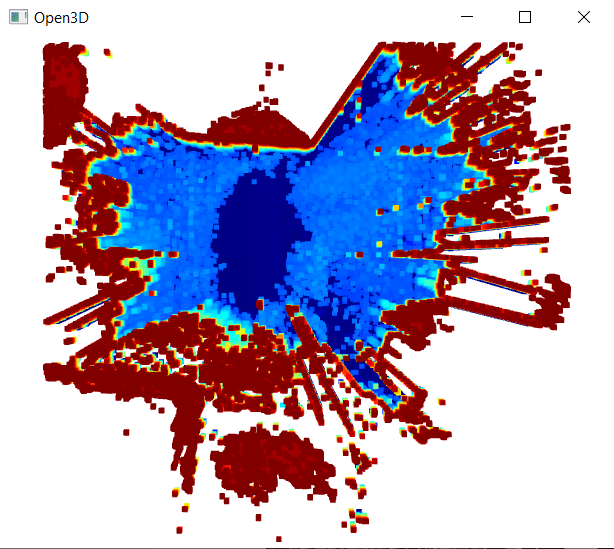


Fig. 24. Open3D Visualizer.

## Map Tracking Viewer

The Map Tracking Viewer reads the robot position and orientation from the text file to simulate the real-time robot movement. Then the robot position and orientation are sent to the backend services. After retrieving the 2D Map, Front / Rear view pictures from the backend service, all the view pictures are displayed on the Map Tracking Viewer GUI.

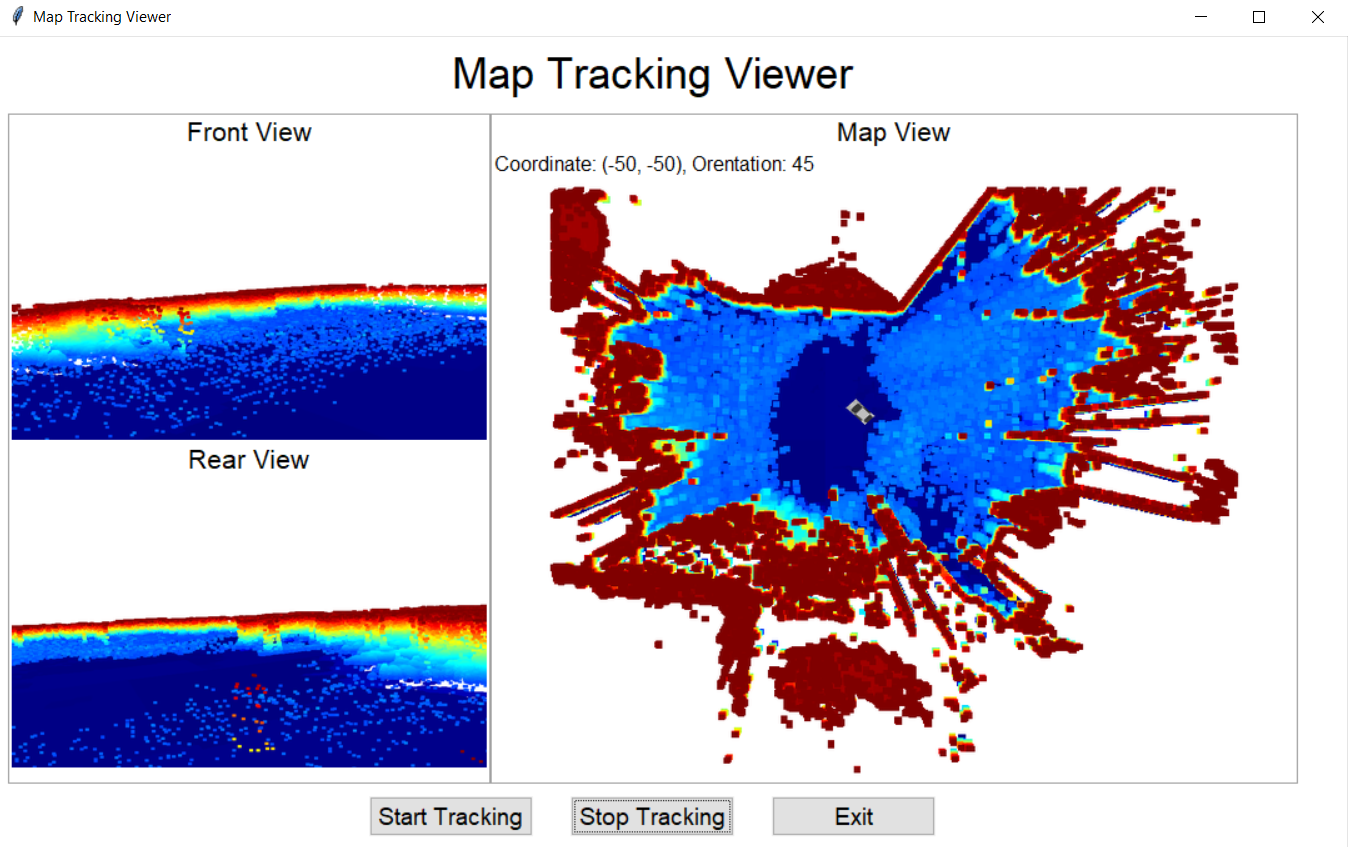


Fig. 25. Map Tracking Viewer.

# CONCLUSION

The Keras neural network can be used to make prediction on the surrounding terrain based on the lidar or point cloud data. With appropriate training with actual lidar scan data, the accuracy is comparable to the existing method such as K-Means method. The output of the Keras neural network can be used for further 2D-3D visualization and allow autonomous vehicle driver to have a better information on the terrain map and predicted view of the surrounding. The visualization system demonstrates the usage of 2D map generated by the Keras neural network.

# LIMITATION AND FUTURE WORK

## Limitation

The Keras neural network currently only processed 1024 x 1024 scope scan data at one time due to GPU power limitation. The processing GPU in use at the project is one Nvidia GTX 1080 graphics card. To improve accuracy and speed of the Keras neural network, more powerful GPU resources are required.

The Keras neural network is not integrated with any existing robot platform such as Robot Operating System (ROS). Therefore, this causes the longer time and more efforts are required to implement a solution based on the Keras neural network.

## Future Work

We implement the technique to split and combine multiple 1024 x 1024 grid size data with reduced edge noises. The technique can be further improved by optimizing with different padding size and grid size.

To reduce GPU power requirement, MobileNetV2 structure is considered to implement the Keras neural network [34][35][36]. MobileNetV2 is light weight deep neural network, contains fewer parameters comparing with traditional ResNet network. MobileNetV2 is designed to work on the mobile platform, it reduces requirement on GPU processing power greatly with acceptable accuracy loss [37]. This will increase the usage of the Keras neural network with reduced GPU power requirement and mobile platform support.

The ROS packages can be created to provide the integration and visualization with the Robot Operating System (ROS). [38] This will reduce the time and effort greatly to implement a solution with ROS development environment [39] and we expect the usage of the Keras neural network will be greatly increased.

References

1. S. Soro and W. Heinzelman, “A survey of visual sensor networks,” Adv.

Multimedia, vol. 2009, pp. 1–22,

1. *D. Lv, X. Ying, Y. Cui, J. Song, K. Qian and M. Li, "Research on the technology of LIDAR data processing," 2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS), Harbin, 2017, pp. 1-5, doi: 10.1109/EIIS.2017.8298694.*
2. *Ravankar, Ankit & Hoshino, Yohei & Emaru, Takanori & Kobayashi, Yukinori. (2012). Robot Mapping Using k-means Clustering Of Laser Range Sensor Data. Bulletin of Networking, Computing, Systems, and Software. 1. pp-9.*
3. *Ahmed, Mahmoud & Guillemet, Adrien & Shahi, Arash & Haas, Carl & West, Jeffery & Haas, Ralph. (2011). Comparison of Point-Cloud Acquisition from Laser-Scanning and Photogrammetry Based on Field Experimentation. Proceedings, Annual Conference - Canadian Society for Civil Engineering. 3.*
4. *Fukushima, Kunihiko (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position" (PDF). Biological Cybernetics. 36 (4): 193–202.*
5. *Nagi, Jawad & Ducatelle, Frederick & Di Caro, Gianni & Ciresan, Dan & Meier, Ueli & Giusti, Alessandro & Nagi, Farrukh & Schmidhuber, Jürgen & Gambardella, Luca Maria. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. 2011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011. 342-347. 10.1109/ICSIPA.2011.6144164.*
6. https://keras.io/api/layers/convolution\_layers/convolution2d/
7. *Sebastian Thrun (2002). Particle Filters in Robotics. Proceedings of Uncertainty in AI (UAI).*
8. *Andreas Nüchter, Kai Lingemann, and Joachim Hertzberg, Hartmut Surmann (2007).* *6D SLAM—3D Mapping Outdoor Environments*. *Journal of Field Robotics 24(8/9), 699–722*.
9. *Shoudong Huang & Gamini Dissanayake (2007). Convergence Analysis for Extended Kalman Filter based SLAM. Robotics, IEEE Transactions on. 23. 1036 - 1049. 10.1109/TRO.2007.903811..*
10. *Inam Ullah & Xin Su & Xuewu Zhang & Dongmin Choi (2020). Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter. Mobile Intelligence Assisted by Data Analytics and Cognitive Computing.*
11. *Sebastian Thrun & Michael Montemerlo (2006). The GraphSLAM*

*Algorithm with Applications to Large-Scale Mapping of Urban Structures. The International Journal of Robotics Research.*

1. *J. B. MacQueen (2006). “Some methods for classification and analysis of multivariate observations,” in Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, L. M. L. Cam and J. Neyman, Eds., vol. 1. University of California Press, 1967, pp. 281–297..*
2. *Forgy, E.. “Cluster analysis of multivariate data : efficiency versus interpretability of classifications.” Biometrics 21 (1965): 768-769.*
3. *Jiahui Huang & Sheng Yang & Zishuo Zhao & Yu-Kun Lai & Shi-Min Hu (2019) “ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation”, IEEE/CVF International Conference on Computer Vision (ICCV)*
4. *Robert R Sokal. A statistical method for evaluating systematic relationship. University of Kansas science bulletin, 28:1409–1438, 1958*
5. *K.Sasirekha & P.Baby (2013) “Agglomerative Hierarchical Clustering Algorithm- A Review”,* *International Journal of Scientific and Research Publications, Volume 3, Issue 3*”
6. *Michael Montemerlo (2003). “FastSLAM: A Factored Solution to the*

*Simultaneous Localization and Mapping Problem with Unknown Data Association”. http://www.cs.cmu.edu/~mmde/mmde-thesis.pdf*

1. *Sebastian Thrun & Michael Montemerlo & Daphne Koller & Ben Wegbreit (2006). “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”. The International Journal of Robotics Research.*
2. *T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In Proc. IEEE Intl. Conf. on Robotics and Automation, pages 424–429, 2006.*
3. *Yiqiao Wang, Wei Zhang, and Pei An, "* *A Survey of Simultaneous Localization and Mapping on Unstructured Lunar Complex Environment" AIP Conference Proceedings 1890, 030010 (2017).*
4. *F. Zhang, S. Li, S. Yuan, E. Sun and L. Zhao, "Algorithms analysis of mobile robot SLAM based on Kalman and particle filter" 2017 9th International Conference on Modelling, Identification and Control (ICMIC), Kunming, 2017, pp. 1050-1055, doi: 10.1109/ICMIC.2017.8321612.*
5. *Aulinas, Josep & Petillot, Yvan & Salvi, Joaquim & Llado, Xavier. (2008). The SLAM problem: a survey. Frontiers in Artificial Intelligence and Applications. 184. 363-371. 10.3233/978-1-58603-925-7-363.*
6. *Jos´e A. Castellanos, Jos´e Neira, Juan D. Tard´os, "Limits to the Consistency of EKF-based System" 5th IFAC/EUCON Symposium on Intelligent Autonomous Vehicles, 2004*
7. *M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem" Proceedings of the National Conference on Artificial Intelligence, pp. 593–598, 2002.*
8. *M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "* *FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges" 18th International Joint Conference on Artificial Intelligence (IJCAI), Acapulco Maxico, pp. 1151–1156, 2003.*
9. *Brooks A., Bailey T. (2009) HybridSLAM: Combining FastSLAM and EKF-SLAM for Reliable Mapping. In: Chirikjian G.S., Choset H., Morales M., Murphey T. (eds) Algorithmic Foundation of Robotics VIII. Springer Tracts in Advanced Robotics, vol 57. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00312-7\_40*
10. *G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in IEEE Transactions on Robotics, vol. 23, no. 1, pp. 34-46, Feb. 2007, doi: 10.1109/TRO.2006.889486.\*
11. *R.Lemus,S. Díaz, C. Gutiérrez, D. Rodríguez and F. Escobar, "* *SLAM-R Algorithm of Simultaneous Localization and Mapping Using RFID for Obstacle Location and Recognition" Journal of Applied Research and Technology Volume 12, Issue 3, June 2014, pp 551-559*.
12. *Thrun, S.; Burgard, W.; Fox, D. (2005). Probabilistic Robotics. Cambridge: The MIT Press. ISBN 0-262-20162-3.*
13. *Omar Takleh, Talha Takleh & abu bakar, Nordin & Rahman, Shuzlina & Hamzah, Raseeda & Abd Aziz, Zalilah. (2018). A brief survey on SLAM methods in autonomous vehicle. International Journal of Engineering and Technology(UAE). 7. 38-43. 10.14419/ijet.v7i4.27.22477.*
14. *Dmitry Yarotsky. (2018). Universal approximations of invariant maps by neural networks. arXiv:1804.10306.*
15. Yan-Bin Jia *(2020).* *Rotation in the Space. http://web.cs.iastate.edu/~cs577/handouts/rotation.pdf.*
16. *Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen (2018). Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://openaccess.thecvf.com/content\_cvpr\_2018/html/Sandler\_MobileNetV2\_Inverted\_Residuals\_CVPR\_2018\_paper.html*
17. *A. G. Howard, M. Zhu, B. Chen et al. (2017). Mobilenets: efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861.*
18. *Yibin Huang,Congying Qiu,Xiaonan Wang,Shijun Wang,and Kui Yuan (2020). A Compact Convolutional Neural Network for Surface Defect Inspection. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7180441/.*
19. *Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano (2017). Benchmark Analysis of Representative Deep Neural Network Architectures. https://arxiv.org/pdf/1810.00736.pdf*
20. *Brian P. Gerkey (2020). acml Package Summary. http://wiki.ros.org/amcl*
21. *Tim Stahl, Alexander Wischnewski, Johannes Betz, and Markus Lienkamp (2019). ROS-based localization of a race vehicle at high-speed using LIDAR. https://www.e3s-conferences.org/articles/e3sconf/pdf/2019/21/e3sconf\_icpeme2018\_04002.pdf*
22. *Zhou, Qian-Yi et al. “Open3D: A Modern Library for 3D Data Processing.” ArXiv abs/1801.09847 (2018): n. pag.*