Point Cloud Compression for 3D LiDAR Sensor using Recurrent Neural Network with Residual Blocks

Chenxi Tu, Eiriro, Takeuchi, Alexander Carballo, Kazuya takeda

12/05/2021

2019 IEEE International Conference on Robotics and automation (IEEE ICRA)

Abstract

- Due to the sparseness and disorderly nature of this data, it is difficult to compress it directly into a very low volume.
  - A potential solution is utilizing raw LiDAR data
- Rearrange the raw data from each frame losslessly in a 2D matrix, making the data compact and orderly
- Uses a recurrent neural network and residual blocks to progressively compress one frame's information from 3D LiDAR
- Describe how decompressed point cloud data can be used with SLAM (simultaneous localization and mapping) as well as for localization using a given map

Contents

Introduction

Motivation

1. It is hard to compress point cloud to very low volume directly using generic methods

2. As 2D matrices of raw LiDAR packet data are irregular, its PCC results :
   a. Sacrifice accuracy
   b. Reduce number of bits

Main contribution

The main contributions of this paper are as follows:

- An RNN to compress a point cloud from a 3D LiDAR,
- A new decoder with a residual block structure, with increased accuracy.
- Evaluation of how decompressed point cloud data works in SLAM(simultaneous localization and mapping) and localization tasks

Related Work

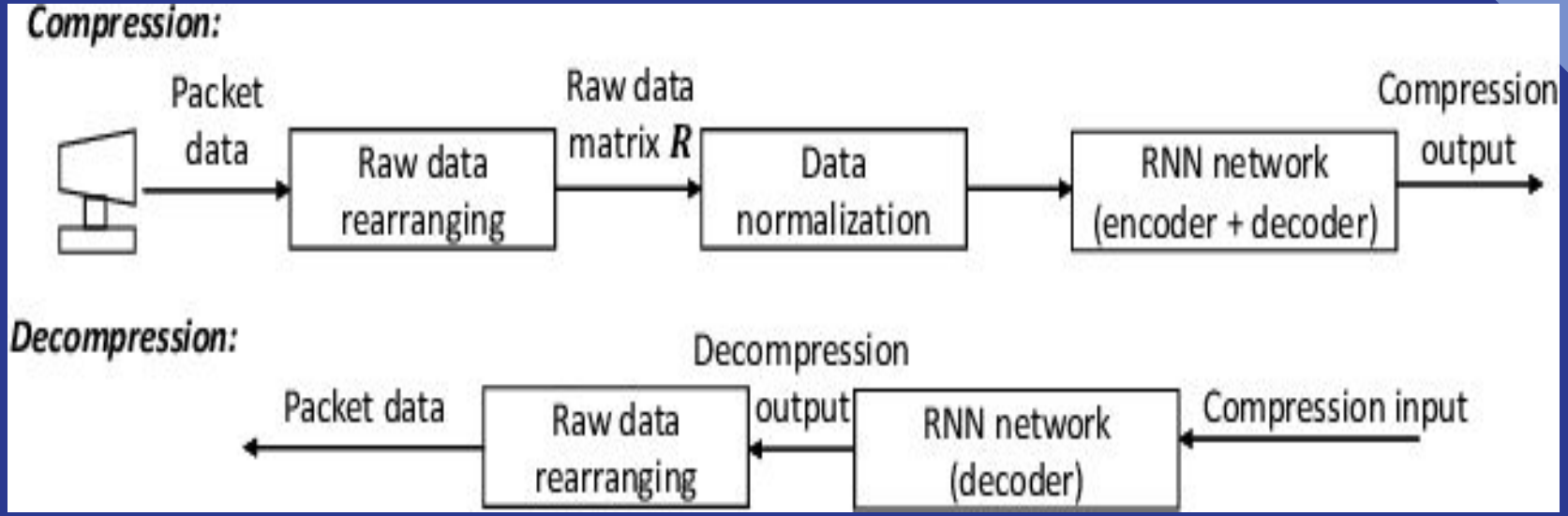
Yin and Berger [1]

- Directly converting a 3D point cloud into a 2D matrix
- Lossless PCC
- Sacrificed the accuracy of the distance and yaw angle information

C. Tu, E. Takeuchi [2]

- Used raw packet data from LiDAR
- Is a highly cost-effective approach for compression
- 2D matrix does not have any direct spatial correlation

Method



Flowchart of proposed compression method

Raw data rearranging

1. 3D LiDAR information stored in 2D matrix ***R*** like an image [1].
2. For each pixel $R_{i,j}$ in ***R***
   a. Laser ID information ***i***
   b. Rotation angle ***j***

is coded into few bits of data losslessly with a few additional bits.

3. Resulted data is irregular and don't have spatial correlation

Data normalization

Before training:

1. Randomly choose packet data frames
2. Calculate mean **μ**
3. Create histogram of distance values
4. Normalize data matrix ***R***

$$(R - \mu) / \theta$$

Network Structure

Single iteration of network is represented as:

$$b_t = B(E_t(r_{t-1})), \quad \hat{x}_t = D_t(b_t) + \hat{x}_{t-1}, \tag{1}$$

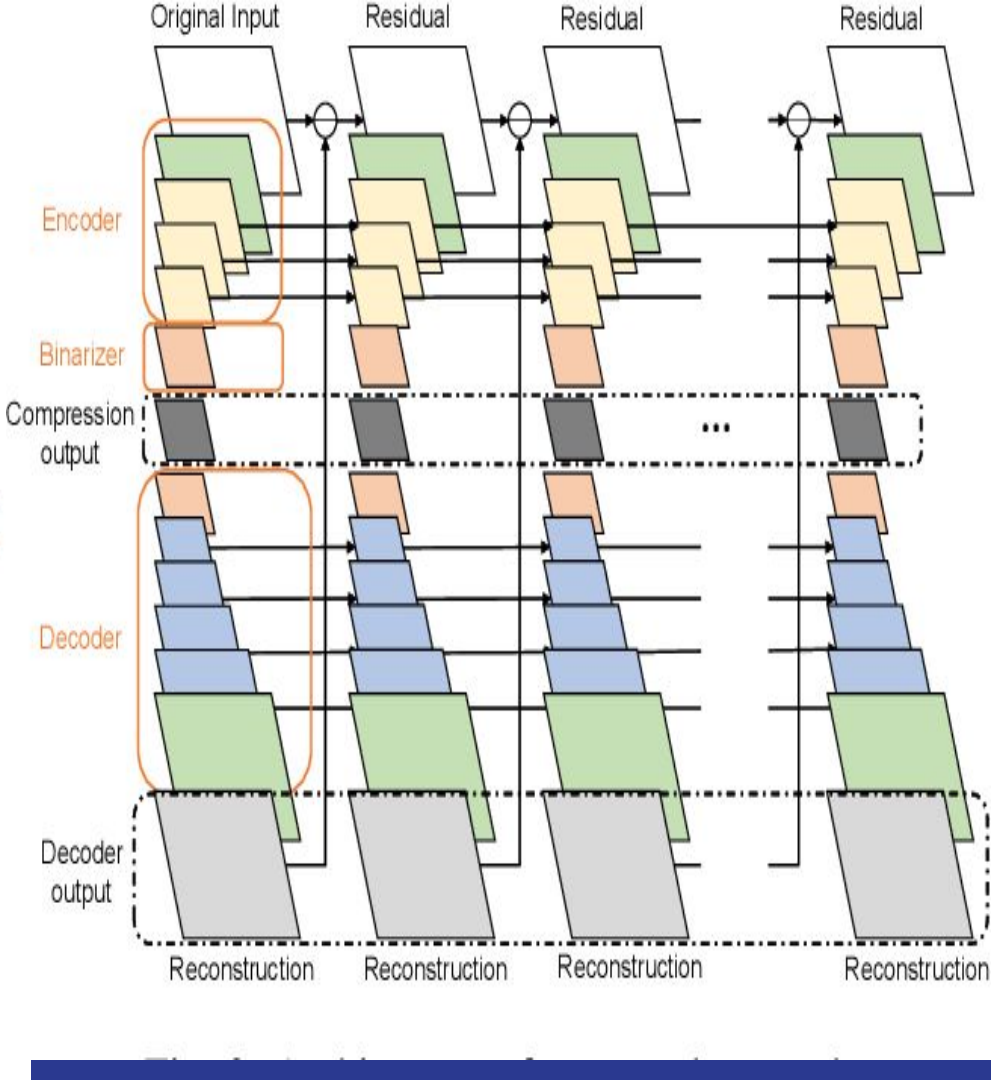
$$r_0 = x, \quad \hat{x}_0 = 0 \tag{2}$$

Where;

$$b_t \in \{-1, 1\}^m$$

With average residual loss of network;

$$\frac{1}{t}\sum_t |r_t|$$



Encoder

1. One convolutional layer & three convolutional LSTM layers
   a. With convolutional operator instead of matrix multiplication

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
\tag{4}
$$

With sigmoid activation function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Binarizer

Binarizer follows encoder and consists of:

- One ***convolution*** *layer*
  - *Outputs 1x1x32 array*
- A ***tanh*** function
- A ***sign*** function

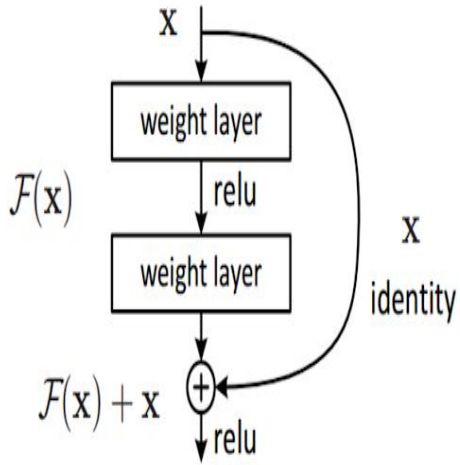
After each iteration 32x32x1 input array reduced to 1x1x32 binarized representation per iteration

- Resulting in **⅛** bit per point.

Basic decoder

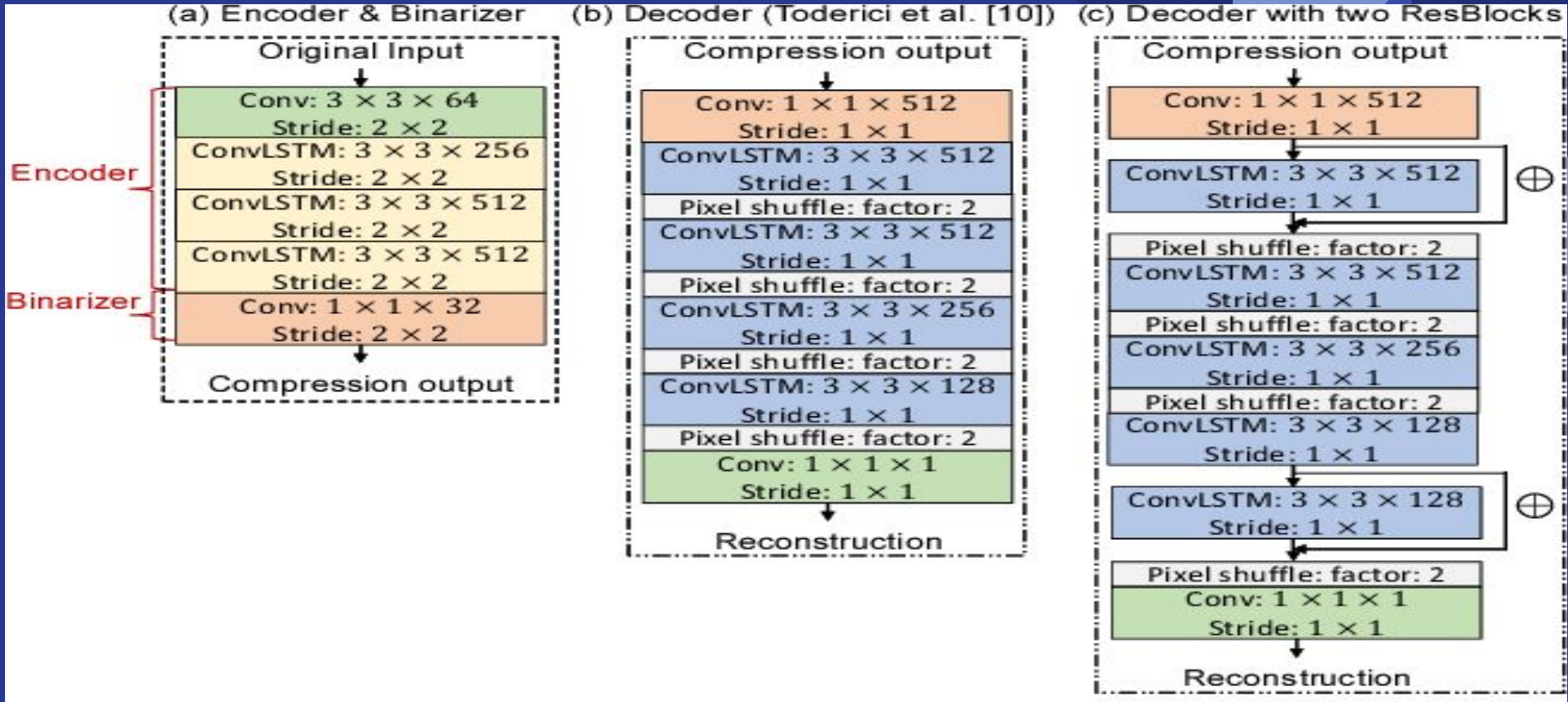
To reconstruct 32x32x1 input from 2x2x32 basic decoder is used

- A convolutional layer expands the input channel
- A convolutional LSTM layer & pixel shuffle layers
  - Used to double the size of input sensor (superresolution task[1])
  - 4 pixel shuffle layers for reconstruction of same output as original input



Decoder with residual block

- Adding more layers in basic decoder will not improve quality in some scenarios resulting in building residual Network [2]
- Due to time constraints only two residual blocks are used in this work



Network details

Evaluation

Setup

Training:

a. 33134 frames
b. 1Hz sampling of driving data
c. 11 areas of Japan
d. Velodyne HDL-32E sensor

Testing:

a. 32 min driving data of Akagi
b. Extracted 32 frames by 1/60 Hz sampling

A. Comparative evaluation

Given two point cloud frames **P** ; **Q**, for each point p ∈ **P** find the closest (Euclidean distance) point q∈ **Q**, q = NN(p; **Q**)

The calculated error is;

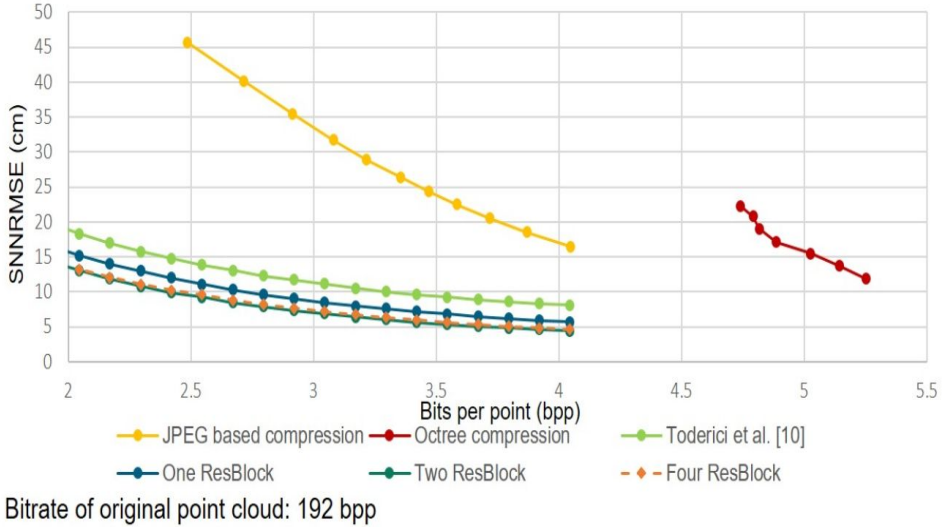
$$\mathrm{MSE}_{\mathrm{NN}}(\boldsymbol{P},\boldsymbol{Q}) = \sum_{p\epsilon\boldsymbol{P}}(p-q)^2/|\boldsymbol{P}|$$

where |**P**| represents the number of points in **P** :

$$\mathrm{RMSE}_{\mathrm{NN}}(\boldsymbol{P},\boldsymbol{Q}) = \sqrt{\mathrm{MSE}_{\mathrm{NN}}(\boldsymbol{P},\boldsymbol{Q})}$$

Symmetric Nearest Neighbor Root Mean Squared Error (SNNRMSE)

$$\mathrm{SNNRMSE}(\boldsymbol{P},\boldsymbol{Q}) = \sqrt{0.5\mathrm{MSE}_{\mathrm{NN}}(\boldsymbol{P},\boldsymbol{Q}) + 0.5\mathrm{MSE}_{\mathrm{NN}}(\boldsymbol{Q},\boldsymbol{P})}$$

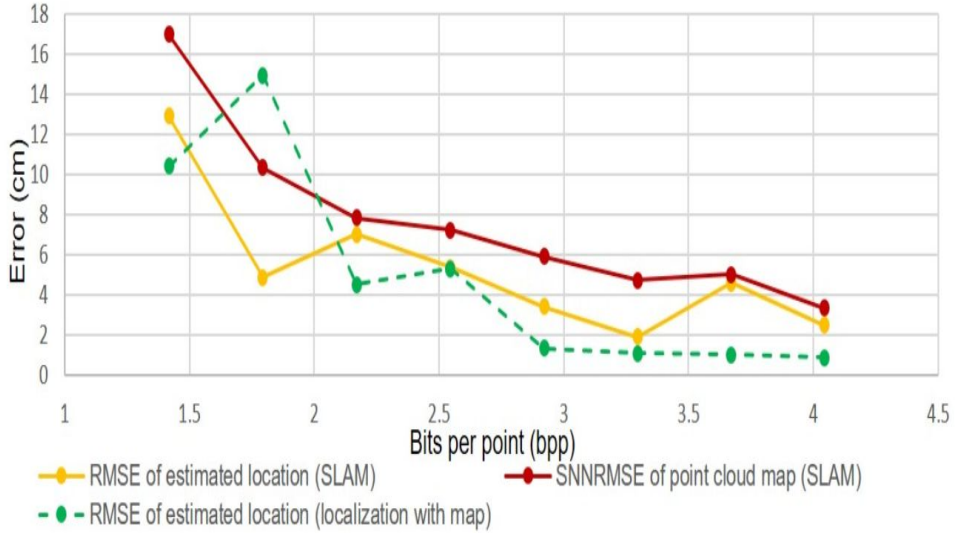


Bitrate of original point cloud: 192 bpp

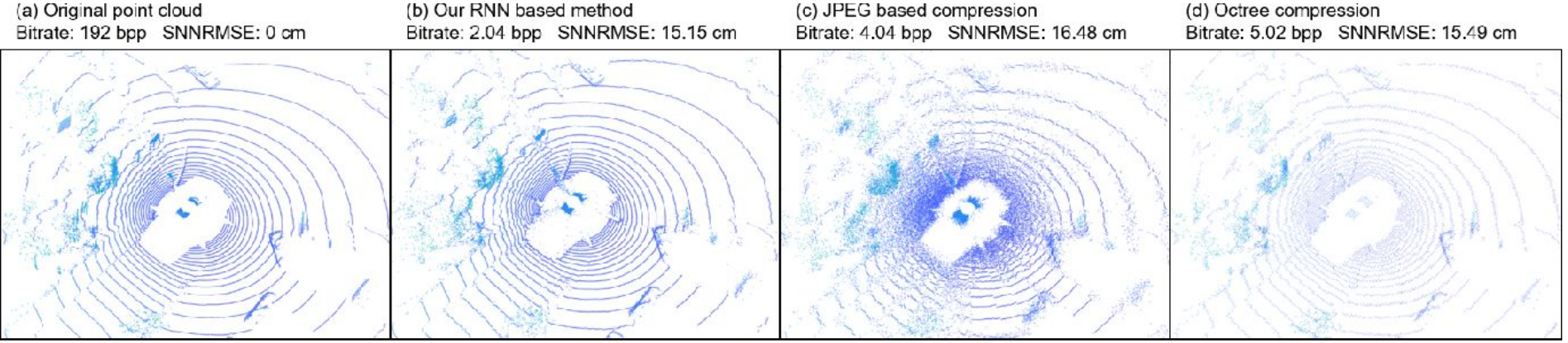
Comparison of various point cloud compression methods

B. Evaluation by application

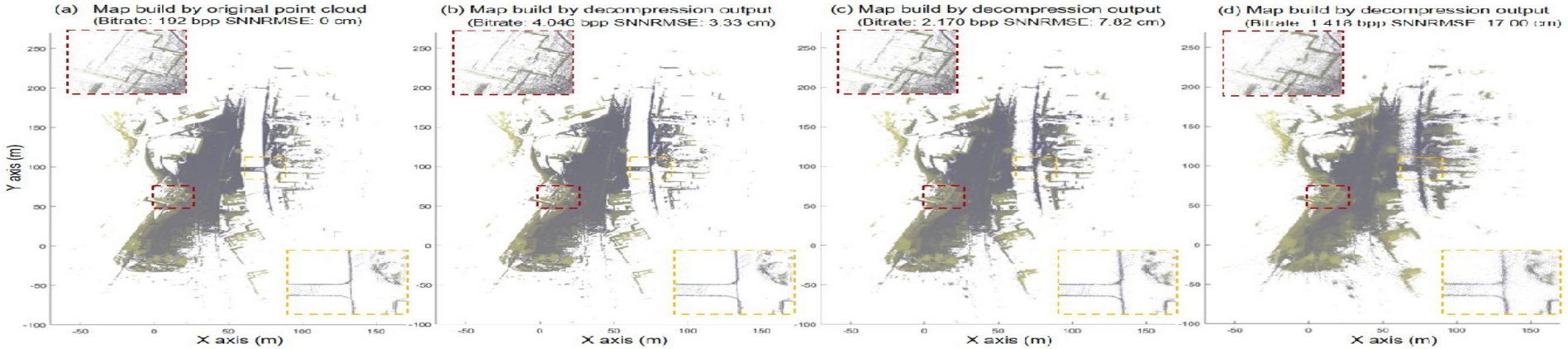
a. Two applications
   i. SLAM
   ii. Localization



Evaluating decompression result by applications



Example of original point cloud and decompressed point cloud from various approaches (colored by height).



Visualizing point cloud maps built by SLAM using original point cloud and decompressed point cloud at various compression rate (colored by height)

Conclusion

Final points

1. Proposed the use of a RNN to compress PC data from 3D LiDAR.
2. Proposed method tune the compression rate with decompression error.
3. this paper shows the potential uses of the proposed method in real robotics applications

Thanks!

Contact :

Muhammad Asad

IIS-Academia Sinica, Taipei TAIWAN

muhammad.asad.uol@iis.sinica.edu.tw