3D Convolutional Neural Networks for Landing Zone Detection from LiDAR

Daniel Mataruna and Sebastian Scherer

Presented by: Sabin Kafle
Outline

● Introduction
● Preliminaries
● Approach
  ○ Volumetric Density Mapping
  ○ Volumetric Convolutional Neural Nets
● Experiments
  ○ Datasets
  ○ Results
Introduction

- A system to detect a Landing Zone for autonomous helicopters is developed using 3D convolutional neural network and processed LiDAR point cloud as input.
- The system outperforms various benchmarks for small and potentially obscured obstacles detection in vegetation terrain.
System

1. Helicopter scanning a candidate landing zone.

2. Point cloud conversion to density map.
   \[ \text{Prob}(\text{safe} | \mathbf{x}) = 0.2 \]

3. 3D CNN processing.
Key challenges

- Engineering a system capable of reliably detecting landing safe site is difficult
  - Appearance of vegetations, obstacles is heterogeneous and difficult to capture with hand build rules
  - Noisiness and sparsity of point cloud data
- Machine learning system also has issues
  - Maintenance of relevant information from large stream of data
  - Obtaining Labelled data (is an expensive and laborious process)
Solutions offered

- Use 3D CNN with volumetric occupancy map which can perform real time safety evaluation once trained
- Use semi-synthetic point clouds to generate labelled data
Preliminaries

- LiDAR
- Convolutional Neural Network
LiDAR

- A remote sensing technology
- Measures distance by illuminating a target with laser and analyzing the reflected light
- Popular used technology to make high-resolution maps
- A narrow laser-beam can map physical features with high resolutions;
  - An aircraft can map terrain at 30cm resolution or better.
Fully-connected Neural Network (NN)
Convolutional Neural Networks
CNNs
Example Architecture

- **INPUT** [32x32x3] will hold the raw pixel values of the image
- **CONV** layer will compute the output of neurons that are connected to local regions in the input
- **RELU** layer will apply an elementwise activation function, such as the max (0,x) thresholding at zero.
- **POOL** layer will perform a downsampling operation along the spatial dimensions
- **FC** (i.e. fully-connected) layer will compute the class scores
Example
A single layer example

- An example input volume in red (e.g. a 32x32x3 image), and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially.
Examples

- A CNN Weight Update
  - Weight Update
- A CNN demo on MNIST data
  - MNIST
Approach

- Volumetric Density Mapping
- Volumetric CNNs
Volumetric Density Mapping

- The volumetric map covers a horizontal area of 100m$^2$ to 200m$^2$ and a height of 2 to 4m.
- CNN is used to independently predict the safety of 1m$^3$ subvolume within this map.
- Space containing vegetation is “porous”
  - Probabilistic formulation of this idea is used
- For a given grid cell coordinate (i,j,k), T sequence of point data that either hit (z = 1) or pass through the grid cell (z = 0) is used
  - A voxel size of 0.05m$^3$ is used
  - For each cell state, Beta Process is used with parameters (alpha and beta) initialized to 1 and updated as
    - Alpha = Alpha + z
    - Beta = Beta + (1 - z)
  - Posterior mean is calculated based on
    - (Alpha) / (Alpha + Beta)
Volumetric Density Mapping (contd..)

- Non-overlapping tiles of 1m$^2$ along XY-plane are generated
  - Ground surface height for each tile is obtained using minima calculation of height of all occupied cells within
  - Then 1m$^3$ volume is placed on each tile with center along ground surface height.
  - Each grid has resolution of 0.05m$^3$, so each subvolume has dimension of 20 x 20 x 20.

- TO SUMMARIZE
  - Mean calculated using Beta variable parameters are considered as pixel intensity of each individual sub-volume
  - A pixel for this mapping is of volume 0.05m$^3$
  - The dimension of resulting CNN will have input of size 20 x 20 x 20.
Volumetric Convolutional Neural Network
Model Parameters

- Hyperparameters
  - Number of convolutional layers
  - Number of feature maps
  - Dimension of features (dimension of local connectivity)
  - Pooling Dimension

- Parameters
  - Weight and Bias of each neurons (including convoluted and fully-connected)

- Theano Library is used for computing gradients and Training
Experiments

- Datasets
- Results
Datasets

- **Synthetic Datasets**
  - Scanning pattern of LiDAR sensor is simulated by its pulse repetition rate, angular resolution and sensor mount behavior
  - Gaussian Noise is added to range (based on manufacturer's specification)
  - Scenes were scanned until point density of 3000 points/m² was reached
  - **First Synthetic Dataset**
    - Ground surface is a mesh with height perturbed using noise
    - Grass blades are simulated using three triangle strip
    - Grass placement is generated using Poisson process
    - Box obstacles are placed
  - **Second Synthetic Dataset**
    - Box obstacles are replaced with 3D model selection
      - 11 models of rock are used along with a tire and cinder blocks
  - For each dataset, various parameters are swept, with 20 (first) instances and 40 (second) per parameter setting
  - For each case, only half have an obstacle.
  - Finally, generated data are augmented after being perturbed (to learn invariant classifier)
Datasets (contd..)

- Semi-synthetic Data
  - Consists of real point cloud data for vegetation and ground combined with simulated data for obstacles
  - Can be achieved simply by inserting virtual obstacles into the world frame and altering rays if they interfere
  - Noise are added into rays in case of interference
Results

- **Evaluation Metric**
  - Receiver Operating Characteristic (ROC) curves are used to compare different algorithms

- **Baselines**
  - Random Forest Classifier
  - Residuals of a robust plane fit
  - Feature Bag of Words (BoW) [until recently considered state of art]

- **Experiments**
  - The first synthetic data set is reserved to chose hyper-parameters for CNN. Two CNNs were chosen for final comparison
    - C7-F64-P4-D512
    - C7-F32-P2-C5-F64-P2-D512
Results

Fig. 8. ROC curve on the second synthetic dataset.
Results

Fig. 9. ROC curve on the semi-synthetic dataset.
Misclassification

Fig. 11. Top: two correct assessments (true positive and true negative). Bottom: Two failures (false negative and false positive). Obstacles are shown in red. In the first failure case, there are several dense bushes which are similar to rocks. In the second, only a very small portion of the obstacle is visible.
Future works

- Extend model to include more information including intensity, other source data etc.
- More extensive evaluations
  - Current model might be overfitting to surface with grass
Questions?

THANK YOU
References

- Voxnet [https://github.com/dimatura/voxnet](https://github.com/dimatura/voxnet)