3D Object Recognition based on Volumetric Representation using Convolutional Neural Networks

Xiaofan Xu[1,2], David Corrigan[2], Alireza Dehghani[1], Sam Caulfield[1,2], and David Moloney[1]

[1] Movidius Ltd. 1st Floor, O'Connell Br House, D'Olier St, Dublin, Ireland. {firstname.lastname}@movidius.com

[2] Trinity College Dublin, College Green, Dublin, Ireland. corrigad@tcd.ie

Abstract. Following the success of Convolutional Neural Networks on object recognition and image classification using 2D images; in this work the framework has been extended to process 3D data. However, many current systems require huge amount of computation cost for dealing with large amount of data. In this work, we introduce an efficient 3D volumetric representation for training and testing CNNs and we also build several datasets based on the volumetric representation of 3D digits, different rotations along the x, y and z axis are also taken into account. Unlike the normal volumetric representation, our datasets are much less memory usage. Finally, we introduce a model based on the combination of CNN models, the structure of the model is based on the classical LeNet. The accuracy result achieved is beyond the state of art and it can classify a 3D digit in around 9 milliseconds.

Keywords: 3D object recognition, volumetric representation, 3D digit dataset, CNN

1 Introduction

Object Recognition (OR) is widely used in our daily life for the purposes of inspection, registration, and manipulation [1]. The well-known applications such as Google, Facebook and Baidu are probably the most famous websites which use OR on a large scale. Generally, object classification is performed using colour based segmentation methods or from grayscale images using classification methods such as HoG[2]/SVM[3] or other classifiers. However, nowadays deep learning is becoming ubiquitous. We are now able to solve some of the problems once considered impossible in fields such as computer vision, natural language processing, and robotics with recent advancements in deep learning algorithms.

Machine learning techniques use data (images, signals, text) to train a model (or machine) to perform image classification or object detection. Although classical machine learning techniques are still being used to solve challenging image classification problems, they don't work well when applied directly to images.

This is because they ignore the structure and compositional nature of images. The state-of-art CNN technique which is a specific type of deep learning algorithm, addresses the gaps in traditional machine learning techniques. CNNs not only perform classification, but they can also learn to extract features directly from raw images which eliminates the need for manual feature extraction. The performance of deep learning which uses CNNs has rapidly grown to over 95% accuracy (GoogLeNet [4], VGG [5], AlexNet [6] etc.) in recent years with the availability of large labelled datasets and powerful GPUs. These methods, while appealing from an accuracy standpoint, are computationally extremely intensive requiring up to hundreds and millions of multiplications per classification. Baidu has achieved the best result to date in the ImageNet classification challenge [7] with custom-built supercomputer called Minwa [8] . Minwa contains 72 powerful processors and 144 GPUs, and it has 6.9TB of host memory, 1.7TB of device memory. These high computational requirements for existing CNN models, such as the Baidu winning model, makes it impossible to deploy CNNs onto mobile devices without modification.

Following the success of CNNs on OR using 2D images [4,5,6]; in this work we extended the framework to process 3D data. Although architectures with volumetric convolutions have been successfully used in video analysis [9,10] where time acts as the third dimension, the nature of our data is very different conceptually. In our proposed approach, CNN classifiers are trained to recognize 3D objects from complete or partial 3D volumetric representations. Many online catalogues of such information are already available which have been generated using 3D scanning devices such as Microsoft's Kinect, or designed using packages such as Blender, 3DMax, Maya or even Minecraft. Unlike the other existing 3D CNNs using 3D point cloud data as training data [11], our dataset is more effective and efficient for training CNNs. The core-structure of our 3D objects is called Volumetric Accelerator (VOLA) which uses a bit-per-voxel representation where the bit corresponding to voxel would be 1 if the volume is occupied by the object and 0 if it is free space. VOLA itself has been shown to offer substantial memory savings compared to other common methods of representing volumetric data, with savings up to 95% for realistic scenes. Based on this, VOLA representation offers the potential to drastically reduce the computational complexity of a CNN as CNNs require huge numbers of multiplications and VOLA can make them trivial as it only requires multiplications by 1 or 0. Therefore, our trained CNN models will be much easier to deploy onto any embedded platform and the computational requirement for classification will be incredible small. For low-cost robotics, the computational complexity and speed of operation are essential for safe interoperation with environments, humans and animals.

The rest of the paper is outlined as follows: Section 2 explains the several datasets have been created for this project along with the CNN models developed based on the proposed datasets. Performance evaluation of CNN models including their accuracy and classification run-time will be presented in Section 3. Finally, conclusions and feature works will be discussed in Section 4.

2 Technical Content

In this section we introduce the datasets structure. A new way to represent volumetric data is explained. The procedure required for generating the dataset is also included.

2.1 Dataset Structure

There are several datasets used in this work in order to evaluate the performance of the CNN models based on volumetric representation. The datasets have been generated using several approaches.

3D MNIST datasets The 3D MNIST dataset has been built based on the original well-known 2D MNIST dataset using MATLAB. It is built in order to train the 3D CNN, performance of which can be compared to the existing LeNet [12] trained on images. Different rotations of the 3D MNIST along x, y, z axes respectively are also included in the dataset as in 3D space rotation is an important component. The degree of rotations was randomly generated between 0° to 360° along 3 axes. Each of these datasets contains a training dataset and a test dataset. Furthermore, the training dataset contains a total number of 59667 3D digits and the test dataset contains a total of 9957 3D digits.

3D-Fonts dataset The 3D Font dataset has been created based on the free fonts available online using Blender. Some examples of the 3D digits are shown in Fig. 1. Specifically, the training dataset contains 256 different fonts and the test dataset contains 79 different fonts. The fonts in the training and test datasets are completely different. The reason for building this dataset is to build a CNN model which can recognize the artificial digits in the real world scenario i.e. 3D digit candles. Similar to the 3D MNIST dataset, separate datasets have been created based on different rotations along the x, y, and z axes and the step of every rotation used in this dataset is 10°. The training set contains a total number of 91440 3D digits and test dataset contains 27000 3D digits. A dataset of compound rotations between x, y and z axes are also created based on 25 different fonts for training and 5 different fonts for testing. The rotation step in the combined rotation dataset is 40° for x, y and z axes because the 3D digits with smaller rotation steps would create a huge dataset with similar performance. This combined rotation dataset has a training set of 144000 3D digits and a test set of 28800 3D digits. What is more, a CNN model based on combinational rotation fonts have been trained to recognize the real world 3D digits. In order to test the CNN model based on these fonts, combinational rotation around x&y, x&z and y&z axes respectively with 10° rotation step was also created. Depth images of different resolutions were captured based on the 3D font dataset. In addition, separate CNN models have been trained using depth images to compare the performance of the 3D Font CNN model and details of the accuracy are included in Section 3.

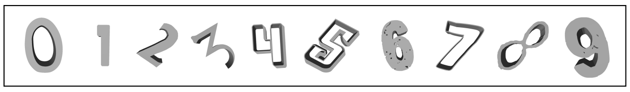

Fig. 1. 3D digits based on different fonts and rotations.

2.2 Volumetric Accelerator (VOLA)

Unlike other existing 3D CNNs using 3D point cloud data as training data [11] or RGB channel along with depth channel for building 3D CNNs [13,14,15,16,17], CNNs in this study can be applied directly to recognize the 3D volumetric representation of the 3D digits. The core-structure of the 3D objects is called Volumetric Accelerator (VOLA). VOLA is a software library for creating, manipulating and visualizing volumetric data. (It is under examination as a patent). More specifically, VOLA deals with regular volumetric grids, also known as voxels and VOLA stores only a single bit of information in each voxel. VOLA uses an octree data structure to store the voxels.

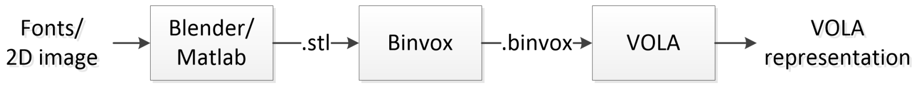

Fig. 2. Steps require for generating VOLA representation.

2.3 Data Format for CNNs

In order to generate the VOLA representation a few steps are required, shown in Fig. 2 below. Specifically, 3D digits with .stl format are converted into .binvox format which is a program that reads a 3D models, rasterizes it into a 3D voxel grid and generates the resulting voxel file. The reason for using VOLA as it only requires 1 bit per voxel, and in .binvox format it requires 1 byte per voxel. VOLA can save more memory usage then the .binvox representation.

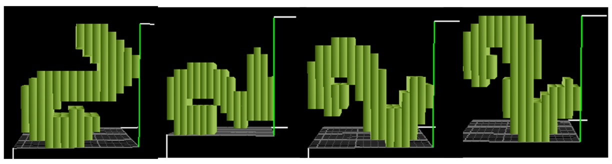

Fig. 3. Digit 2 in binvox format along z-axis rotation with 0°, 40°, 80° and 100° rotation respectively.

Regarding to the structure of the datasets in Section 2.1, different rotations are considered for training and testing the CNN models. Fig. 3 and Fig. 4 show the different rotations in .binvox format. Binvox represents the occupancy in a grid and each voxel in the .binvox represents 1 byte.

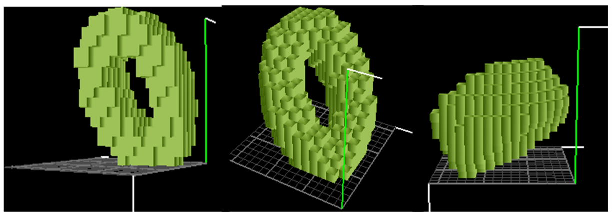

Fig. 4. Digit 0 in binvox format with combinational rotations along x, y and z axes.

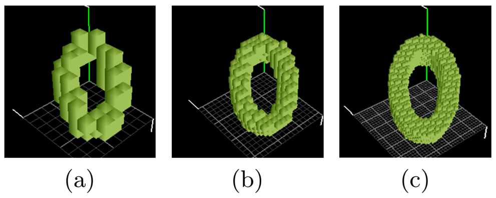

Fig. 5. 3D Same digit 0 with different resolution of binvox:(a) 8×8×8 voxel; (b) 16×16×16 voxel;(c) 32×32×32 voxel.

In this work, all 3D digits are based in a small volume of 16×16×16 voxels. The reason for choosing 16×16×16 voxels is because it gives the best performance in terms of computation and visualization. In Fig. 5, it shows the different resolution of binvox and three different resolutions were tested. In 8×8×8 volumetric grid, most of the useful information of Digit 0 is missing. Although in 32×32×32 volumetric grid the binvox gives the best visualizing output, it would increase the computational requirement for training the CNNs. With 16×16×16 voxels representation, the binvox contains enough useful information for it to be recognized as a zero. Therefore, 16×16×16 volumetric grid was chosen in this work.

After converting the 3D digits to binvox, VOLA can be applied directly to the binvox files. VOLA outputs a binvox file into a single string of 1s and 0s. VOLA starts at point (0, 0, 0) which corresponding to the circle shown in Fig. 6 it moves from right to left along the x axis, bottom to top along the y axis and font to back along the z axis. The output of a single string from VOLA then reshape into a VOLA image which shown in Fig. 6. The intensity of the image contains only 1s and 0s. Each yellow box on the image represent 1 slice of the binvox in X-y plane along the z axis. The size of these yellow boxes are 16×16 which is the size of the x and y dimension in the volumetric grid. These yellow boxes form into a VOLA image which then be used as the training image.

2.4 CNN Architecture

In this work, a few CNN models have been designed based on the famous LeNet model [12] shown in Fig. 7 using CNN framework Caffe [18]. However, the CNN model in Fig. 7 only consists 1 rotation of the 3D digits, the combination rotation

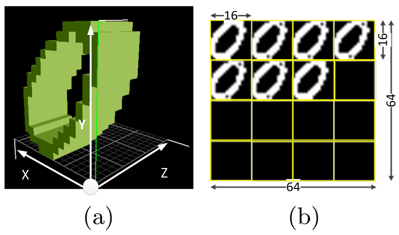

Fig. 6. Binvox to VOLA conversion process:(a) Binvox representation visualization; (b) VOLA representation.

CNN model is described in Section 2.5. Unlike the traditional LeNet model takes in the input of size 28×28, this CNN takes in the input as VOLA image of size 64 × 64 without any resize, the VOLA image is based on the .binvox of size 16×16×16 3D digit. The first convolutional layer contains 20 kernels of size 5×5, and second convolutional layer contains 50 kernels of size 5×5. In the Pool layers, max pooling of 2×2 is applied on each of the feature maps achieved from the convolutional layers. After passing through multiple convolutional layers, max pooling layers and fully connected layer, the 2D VOLA image has been converted into a vector of size 10 contains the probability of the input belongs to each class which can be used for 3D digit classification. All the trainable parameters in each layer are initialized randomly and trained using online error back-propagation algorithm described in [12].

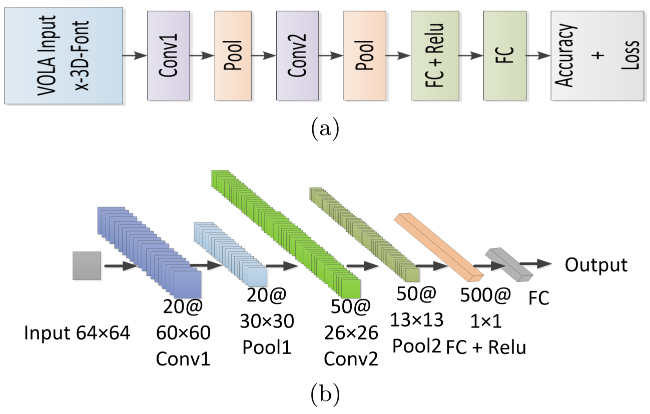

Fig. 7. The CNN model layout for 1 rotation 3D digit recognition. (a) The architecture contains 2 convolution layers, 2 max pooling layers, 2 fully connected layer; (b) The architecture in details.

2.5 Model Combination

Based on the model description in Section 2.4, a combination CNN models is designed for recognize the real world 3D digits. Four CNNs with the structure shown in Fig. 7 are trained based on X-rotation, y rotation, z rotation and xyz rotation 3D digits respectively. Maximum output will be voted as the final output. In real world 3D object contains different rotations, the selection of optimal CNN model for a problem is difficult. Therefore, we construct multiple models and combine the outputs from these models in order to make a best prediction. This has been used in combining traditional neural networks [19]. The input to these models is the same VOLA image. Output results demonstrate that the combination CNN models achieve good prediction on real 3D digit.

3 Experimental Results and Discussion

3.1 Accuracy

The test accuracy based on the test dataset of the individual trained CNNs based on the 3D fonts with different rotations is shown in Table 1. The average accuracy achieved for the 3D font CNN is 81.85%. The reason for having a lower accuracy in z-rotation is due to the fact that Digit 6 and Digit 9 have similar features when these 2 digits rotate in the z direction. Based the 3D MNIST dataset, the average test accuracy reached to 89.14%. The VOLA input to the CNN models are size 64×64 with only 1-bit representation for each pixel. Depth images of size 16×16 with 4-bit representation (16 grayscale level) for each pixel achieved only 63.05% accuracy and achieved 70.04% with 32×32, 6-bit representation per pixel. The details are shown in Table 2. The reason for using this depth image representation is to match the memory footprint with VOLA representation. Low accuracy result for the depth images suggested the features from the 3D shape are important in the recognition process. In order for the depth image to reach the same accuracy level as the VOLA image, it requires image size of 64×64 with 8-bit representation for each pixel. This cause much more memory usage compared to the VOLA image. A separate test was performed based on the xy-plane projection of the 3D digits. However, the accuracy achieved for this case is only 57.85%. This proves that for 3D digit recognition z plane information is important and it also suggests that our proposed CNN models have a good performance.

Table 1. Test accuracy and loss for 3D font CNNs & 3D MNIST CNNs with different rotation.

3D font CNN	**Test Accuracy**	**Test Loss**
X-rotation	82.95%	0.757495
Y-rotation	84.70%	0.774017
Z-rotation	77.33%	1.0494
XYZ-rotation	82.43%	0.700333
3D MNIST CNN	**Test Accuracy**	**Test Loss**
X-rotation	91.80%	0.342894
Y-rotation	92.32%	0.322077
Z-rotation	83.31%	0.738499

Table 2. Accuracy & Loss for depth image.

Depth images	**Accuracy**	**Loss**
16×16 4-bit representation for each pixel	63.05%	0.757495
32×32 6-bit representation for each pixel	70.04%	0.774017
64×64 8-bit representation for each pixel	90.56%	1.0494

3.2 Classification Run-time

We use a Titan-X GPU in the experiments. The performance for classification based on the trained models are listed in Table 3. The results show that for the original LeNet based on 2D MNIST dataset gives slowest classification. It takes 13 msec when classifying 1 image. For the trained 3D digit CNNs, takes around 9 msec to classify 1 3D digit. The results show that our implementation of CNNs based on VOLA representation is fast for classification especially in the convolutional layers. As the VOLA representation made the multiplications more trivial.

Table 3. Classification time for CNN models.

Layers	**3D MNIST X-rotation**	**3D Fonts X-rotation**	**3D Fonts xyz-rotation**	**2D MNIST**
Data	0.050	0.043	0.071	0.064
Conv1	2.207	2.147	2.245	3.343
Pool1	0.133	0.129	0.133	0.063
Conv2	5.841	5.721	5.824	9.759
Pool2	0.070	0.066	0.085	0.036
Ip1	0.379	0.361	0.392	0.157
Relu	0.011	0.009	0.013	0.012
Ip2	0.059	0.049	0.058	0.053
Loss	0.179	0.154	0.179	0.168
Average	0.992	0.964	1.000	1.517
Total	8.929	8.679	9.000	13.655

4 Conclusion

We developed CNN models for recognize 3D digits in this paper and we also created several datasets based on the 3D digits using VOLA representation. The final output of the CNN is obtained by combining the information from all the CNN models based on different rotation of the 3D digits. We also analysis the performance of the CNN models based on our VOLA volumetric representation. The test accuracy we achieved is around 80%. In this paper we only tested the performance based on the test dataset. We can also test our model with the real world 3D digits which can be captured using Kinect or any other devices in the future.

Acknowledgment

This work has been partially supported by Project Eyes of Things (EoT) Grant n. 643924 from the European Union's Horizon 2020 Research and Innovation Program.

References

1. D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.
2. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.
3. C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
5. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
7. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
8. R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," *arXiv preprint arXiv:1501.02876*, vol. 22, p. 388, 2015.
9. S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013.
10. A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.
11. D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.
12. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
13. S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," *arXiv preprint arXiv:1511.02300*, 2015.

14. I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
15. R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, “Convolutional-recursive deep learning for 3d object classification,” in *Advances in Neural Information Processing Systems*, 2012, pp. 665–673.
16. L. A. Alexandre, “3d object recognition using convolutional neural networks with transfer learning between input channels,” in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 889–898.
17. N. Höft, H. Schulz, and S. Behnke, “Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks,” in *KI 2014: Advances in Artificial Intelligence*. Springer, 2014, pp. 80–85.
18. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.
19. L. K. Hansen and P. Salamon, “Neural network ensembles,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 993–1001, 1990.