Using a 3D CNN for Rejecting False Positives on Pedestrian Detection

Francisco Gomez-Donoso, Edmanuel Cruz, Miguel Cazorla, Stewart Worrall and Eduardo Nebot
University Institute for Computer Research, *University of Alicante*. Alicante, Spain.
{fgomez, ecruz}@dccia.ua.es, miguel.cazorla@ua.es
Australian Centre for Field Robotics, *University of Sydney*. Sydney, NSW 2006, Australia.
{s.worrall, e.nebot}@acfr.usyd.edu.au

***Abstract*—Self-driving cars are becoming slowly but surely the future of transport. Nonetheless, in order to achieve fully automatic operation, several challenges are still needed to be tackled. One of the main goals that is currently being pursued is a very accurate scene understanding and object detection. In this regard, the most accurate object detectors are image-based. However, these methods yield critical flaws that make them prone to error in some specific scenarios. For instance, actual objects would be detected in depictions of such objects. The urban environment is strewn with these cases. Namely, in billboards and advertisements. However, most of the self-driving cars feature a lidar that provides 3D perception. This sensor could help to disambiguate the cases mentioned before.**

In this paper, we combine the accuracy of 2D deep learning object detectors with a 3D Convolutional Neural Network (3D CNN) for rejecting false positives on pedestrian detection. First, the object detector provides all the detected pedestrians in the scene, and then the 3D CNN is in charge of rejecting or verify the detections. Our proposal is tested on two well-known publicly available datasets and provides up to 84% accuracy.

***Index Terms*—pedestrian detection, self-driving cars, deep learning, neural networks, CNN training**

I. Introduction

It is undeniable that self-driving cars are the future. In fact, a number of automakers are investing vast amounts of efforts and resources in achieving this goal. Nonetheless, a full level of automation has not been accomplished yet. According to the Society of Automobile Engineer International, there are six different levels of autonomy. The levels range from level 0, which means no automation at all, to level 5, which refers to full automation. In the last level, the vehicle is capable of performing all driving functions under all conditions. There currently are commercial models that implement the level 2 of autonomy. Namely, the vehicle has combined automated function, like acceleration or steering, but the operator must remain engaged with the driving task. The next level implies that the autonomous system is able to understand the environment, and this is the challenge nowadays.

In fact, some automakers such as Ford, Mercedes and Toyota already have full self driving car projects and early prototypes. These prototypes feature a range of different sensors like GPS, lidar range device and HD color cameras to enhance the perception of the environment. However, there are other companies, such as Tesla, that have expressed their will to rely only in cameras to achieve the goal.

This work has been supported by the Spanish Government TIN2016-76515-R Grant, supported with Feder funds. This work has also been supported by a Spanish grant for PhD studies ACIF/2017/243, and BEFPI/2019/039. Salvador de Madariaga grant PRX19/00495 has been supported this work as well.

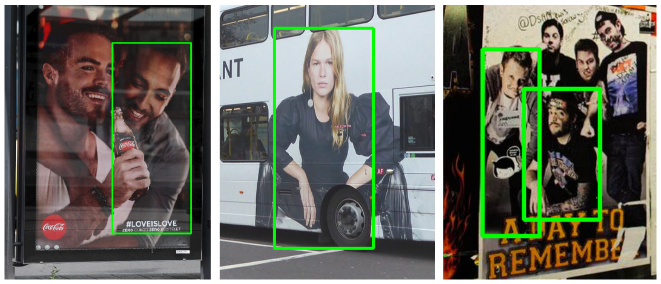

Figure 1: Output of a state-of-the-art object detector. The detections are not correct in a self-driving cars context because they are not actual persons, but depictions of them.

Nonetheless, despite the high accuracy rate of the image-based algorithms, they still have serious functionality gaps. For instance, the methods that work on images rely on visual features to perform classification or segmentation are prone to fail when the target is visually similar to other objects. For instance, an actual person and a poster advert that depicts a person. This effect is specially critic when it comes to self-driving car applications because it cannot confuse an actual pedestrian for an advert. Some of these cases are shown in Figure 1.

In this paper we introduce a pipeline for rejecting false positives by taking advantage of lidar input. Our proposal relies in any image-based object detector to infer the location of the pedestrians, then the corresponding 3D points of each are segmented and forwarded to a 3D Convolutional Neural Network (3D CNN) deep-learning architecture to either confirm or reject the provided prediction. As our proposal uses 3D information, it is able to discriminate between actual objects and other objects that look similar. Our approach is intended to be deployed in self-driving cars that feature color cameras and lidar sensors.

II. Related Works

General 3D object recognition is a blooming research topic nowadays. The most novel approaches that tackle this problem are participants of the ModelNet [1] challenge. This challenge consists on the creation of new 3D object classification methods. To benchmark them, two versions of the dataset are released. First, they released a reduced version with 10 categories and one extended version which consist of 40 categories. The samples that compose the datasets are computer-assisted designed (CAD) models.

Give the fact that there is a challenge, most 3D object classification methods focus on achieving the best accuracy. Due to this, they take advantage of intrinsic features of the data. For instance, in the methods [2], [3], [4], [5], and [6], the authors take advantage of a multi-view approach. All of them use different points of view of the target object in their pipeline. Some of them even render the 3D object to the 2D space by creating a greyscale image of the views.

There also are some approaches that take advantage of the CAD features to recreate a 360 panoramic 2D view of the samples. This is the case of the methods described in [7], [8], or [9]. Once create this 2D representation, they rely on a convolutional pipeline commonly applied to image classification.

The methods discussed so far cannot be applied on a real life use case because the majority of sensors provide point clouds. This representation encodes the tridimensional information they are able to sense. Thus, it is unclear how the mentioned models would perform when fed with pointclouds instead of CAD samples.

Finally, there also are approaches, such as [10], [11], and [12], that take advantage of the pure geometrical features of the data. They rely on an voxelized representation of the tridimensional data so they can create an organized structure in which apply 3D convolutions or other methods.

Nonetheless, the most promising methods do not rearrange or project the point clouds into any intermediate representation. Methods like [13], [14], and [15] run directly on the point clouds to provide classification results.

Regarding the tridimensional object recognition on LIDAR inputs, the problem is even harder because the data is not dense. Namely, the 3D point clouds are more sparse that those provided by stereo setups, structured-light devices or time-of-flight cameras. In addition, this problem is also very related to urban object recognition and autonomous cars use case.

For instance, in [16] the authors propose the projection of the laser data to the ground plane and fed the resulting representation to a 2D convolutional pipeline. Other approaches, such as [17], utilizes the voxelization as a manner to organize the tridimensional space to perform urban object detection and classification. Finally, [18] creates a multichannel discretized 2D representation in which each channel represents a different feature like the occupancy, the range, and the intensity of the laser.

It is not surprising that the majority of the most accurate methods described in this sections rely at some point in a 2D representation. This is due to the fact that the image-based methods achieve a remarkable accuracy in classification tasks. For instance, ResNet50 [19] outperformed the human in the mentioned task on the ImageNet [20] challenge. Nonetheless, the image-based approaches are not completely suitable for urban object and pedestrian recognition due to the reasons discussed in Section I.

As show in this section, the most accurate approaches to 3D object recognition require certain features for the input data that a LIDAR does not yield, such as the ability to be rendered to an image or to provide different views of the objects. In addition, the approaches that do work on actual LIDAR data rely on the discretization of the whole point cloud or in projections to the 2D space, so the topological features are lost. Finally, it is worth mentioning that the majority of these approaches are tested on high-end 64 beams LIDAR devices.

III. Pipeline for Pedestrian Detection

Our approach is intended to be deployed in cars that feature a color camera and a lidar sensor. Furthermore, both sensors must be calibrated so the lidar 3D points can be projected to the image plane. In addition, both sensors must be triggered simultaneously so the output data represent the environment at the same instant.

The pipeline of our proposal is as follows. First, an image is grabbed from the camera. This image is forwarded to any pedestrian detector. At this point it is worth to mention that is preferable to use a fast, greedy detector that provides a great amount of proposals with the aim to include all the pedestrian, even if the majority of the detections are incorrect. After this, a number of areas of interest (AOIs) are produced as a result.

Each AOI represents the potential location of a pedestrian in the image. Then, the corresponding 3D points within each AOI are segmented. Thus, a set of point clouds is obtained, each point cloud representing a potential pedestrian.

Next, a voxel volume is created from each point cloud. To do so, a voxel grid of size $20 \times 20 \times 20$ cells is fit to the size of the point cloud. If a certain cell contains points within it, the corresponding voxel would yield a 1, otherwise it would yield a 0.

Finally, each point cloud is forwarded to the 3D CNN so it states whether the input data represents an actual pedestrian or not.

In the next subsections, the 2D object detector and the 3D CNN are thoroughly described.

A. 2D Object Detection

Despite our proposal could run any pedestrian detector, we choose three different methods which we explain in this

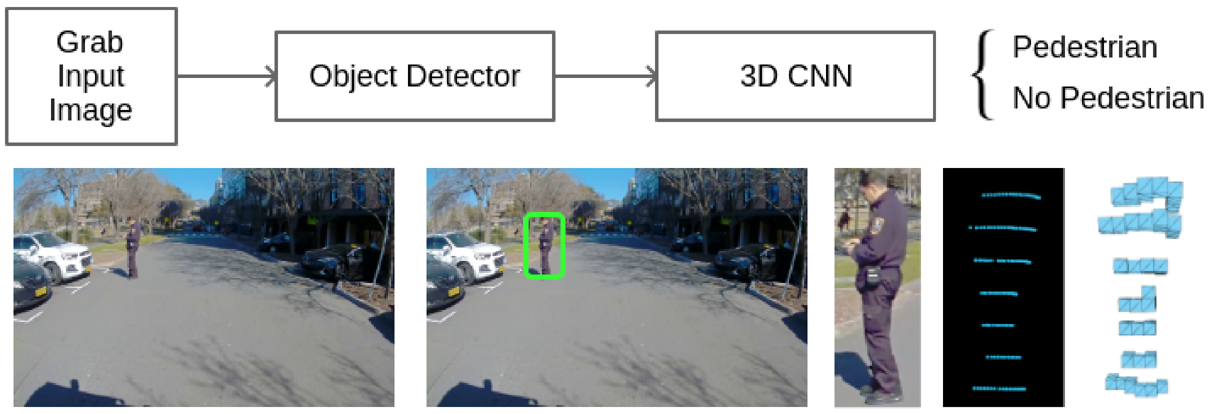

Figure 2: Pipeline of the proposal and intermediate results of each stage.

section.

First, YOLO v3 [21] was chosen because is the most accurate method for object detection. The original model provides detection capabilities for different objects, pedestrians included. YOLO v3 applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. The model has several advantages over classifier-based systems. It looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike other systems which require thousands for a single image. This architecture is one of the best performers on object detection tasks.

Then, we also considered Tiny-YOLO [22]. YOLO v3 explicitly achieves multi-scale learning because its architecture features three region regression networks at three different depth levels. Tiny-Yolo repeats the same YOLO v3 structure up to the first region regression network. This way, it is much faster than the original yolo, but its accuracy level is also weaker.

Finally, we also considered MobileNetSDD. This architecture is a specific implementation of the SSD [23] architecture. The SSD method for detecting objects in images using a single deep neural network. It discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. The SSD model is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network. This method features a convolutional network which can be modified to fit different applications, so we chose MobileNet [24] as the backbone. MobileNet are a set of efficient models called for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. We introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. Thus, as a result of combining SSD and MobileNet we got MobileNetSSD, which is both accurate and fast.

These object detectors are considered because they are remarkable in accuracy or provide a good trade-off between runtime and accuracy. However, any object detector could fit this pipeline.

B. A 3D CNN for False Positive Rejection

The second stage of the proposed pipeline takes as input the proposals of the object detectors, which are the potential pedestrians, and states whether they finally are pedestrians or not. Actually, this piece takes the resulting voxel grid volume corresponding to the 3D points within each 2D AOI.

The 3D CNN of choice is based on the PointNet [25] architecture. This is a pure 3D convolutional neural network that takes occupancy voxel grid volumes and is able to classify them among 10 different classes. The original incarnation of the architecture is intended to work on synthetic datasets or on Kinect-like devices, which provide high density point clouds. Thus, it relies on the density of the input point cloud to populate the cells of the voxel grid volumes. However, our implementation works on LIDAR 3D data, which is greatly sparse.

In order to adequate the network to the current problem, we carried out several modifications to the original PointNet. On one hand, the input voxel volume was reduced from $30 \times 30 \times 30$ to $20 \times 20 \times 20$ voxels in order to better present the 3D data in spite of being much scattered. As a result, the depth of the network was also reduced so the topological features are not lost due to the reducing nature of the convolutions. The last modification consisted on the population of the cells of the input voxel grid volume. In the original PointNet, each cells represented the density of the input point cloud. Nonetheless, we modified that for a binary occupancy grid. In our case, each cell encodes the presence of points in the corresponding 3D space of the input point cloud. Thus, the output of the voxelization process is a volume of $20 \times 20 \times 20$ voxels in which a 1 in a cell represents that the corresponding space yields points and a 0 indicates that there is no points on the corresponding space. Finally, the last fully connected layer was also modified to fit our problem. Namely, the number of output neurons was changed from 10 to 2, which corresponds to the sample being confirmed as pedestrian or not.

Finally, the 3D CNN features the following configuration:

- Input Layer of $20 \times 20 \times 20$ voxels
- Convolution 3D layer featuring 100 filters of size $3 \times 3 \times 3$
- Dropout layer of 50% probability
- Convolution 3D layer featuring 100 filters of size $2 \times 2 \times 2$
- Dropout layer of 50% probability
- Fully connected layer yielding 2 output neurons

C. Dataset and Data Preprocessing

In order to train and test the 3D CNN included in the proposal, the University of Sydney Campus (USYD) Dataset was chosen. This dataset is composed of data provided by different sensors. It provides 16 beams LIDAR readings, 360° view produced by 6 color cameras, very accurate GPS position and inertial data. The coordinate frames of the different sensors are calibrated so it is trivial to transform data to any other sensor. All the mentioned devices are mounted in an electric car, which was driven within the USYD campus once a week per a year. Thus, this dataset features a range of different driving conditions such as crowded roads, pedestrian and vehicle shared spaces, and underground parking lots. It also depicts different lightning and weather conditions. It is worth noting that our approach only takes advantage of the LIDAR and the front color cameras.

The dataset does not provide object labeling, so we used YOLO v3 to automatically extract them. This is a common procedure in machine learning known as pseudo-labeling [26]. Actually, we run the pipeline described earlier until the voxelization procedure. We only considered the odd-numbered sequences of the USYD dataset sequence from 1 to 19. As a result, we extracted the pedestrian training samples. However, to populate the no pedestrian category, we randomly create 5 artificial AOIs in the input images. Despite having random size and random localization within the image, they are required to not to intersect with a pedestrian detection. Then, each AOI is processed the same way the pedestrian AOIs.

Finally, the dataset is composed of more than 24000 pairs of image samples and corresponding point clouds. The dataset is balanced so both categories yield the same number of samples. It was split following the 70% - 30% for training and testing methodology.

At this point it is worth mentioning that the considered action range is $10m$ because the 16 beams LIDAR sensor does not provide enough 3D info above that threshold. Namely, above 10 meters the points are too sparse to be used in our method. Thus, each sample that is above this $10m$ threshold is discarded.

D. Training Procedure

The 3D CNN architecture was trained for 5000 epochs. The optimizer of choice was Adam [27] set up with an initial learning rate of 0.000001. The best ratio between loss and accuracy was provided at epoch 557 so this model was selected for the experiments. After that epoch, the model started to show overfitting issues. Finally, the training loss and accuracy were 0.1297 and 0.95, and the testing loss and accuracy were 0.2381 and 0.92.

IV. Experimentation

The experiments focus on the accuracy of the 3D CNN for false positive rejection. First, YOLO v3, Tiny-YOLO and MobileNetSSD are used as pedestrian detectors. Then, the 3D CNN is used either to confirm that the detected object is a pedestrian or not. We run our algorithm in the selected datasets and stored the prediction results. Then, 100 samples labeled by our system were randomly selected and a human agent stated whether the prediction was correct or not. The ratio of correct predictions corresponds to the accuracy values reported in this section.

The hardware in which the experimentation was carried out consist of an Intel Core i7-8700 CPU @ 3.2 GHz. The setup also features 2 x 16 GB of DDR4 RAM running at 2400 MT/s. An NVIDIA GTX 1080TI was used for speeding up the training and inference processes. The motherboard is an Asus Z390 AORUS ELITE-CF powered by a 1000W PSU. The operating system of choice is Ubuntu 18.04.03. The architectures are implemented using Darknet, Tensorflow 1.14 and Keras 2.3.1. The frameworks take advantage of CUDA 10.2 and cuDNN 7.6.

A. Results on the USYD Dataset

The sequence corresponding to the week 23 of the USYD Campus dataset was used for testing the algorithm in this experiment. It is worth noting that this sequences has not been used in the training procedure. The 10m distance limitation is

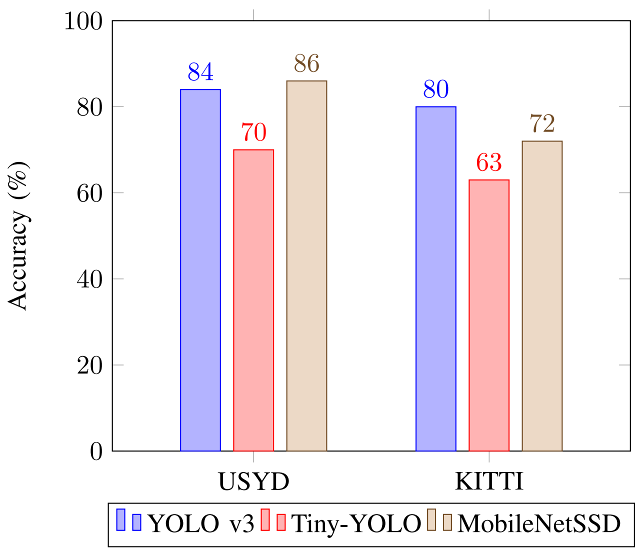

Figure 3: Results of the experiments benchmarking the accuracy of the 3D CNN on different object detectors and different datasets.

kept, so the objects detected above that threshold are ignored.

As shown in Figure 3, our proposal achieved a 84% accuracy when YOLO v3 was involved in the pipeline. Then, if the pipeline runs the Tiny-YOLO object detector, the accuracy dropped to 70%. Finally, the accuracy is 86% when MobileNetSSD is involved as the object detector.

B. Results on the Kitti Dataset

In the sake of comparison, we also put to test our algorithm with the Kitti dataset, which is a state-of-the-art dataset for urban object detection and is commonly used to benchmark the approaches related to self-driving cars. It is worth noting that we removed the 10m constraint in this experiment. Thus, all objects that have corresponding 3D data are fed to the 3D CNN.

Regarding the accuracy when using YOLO v3 as the object detector, the accuracy is 80%. It decreased to 63% when Tiny-YOLO was used in the method. Finally, MobileNetSSD threw an accuracy of 72%.

V. Conclusions

First, we can conclude that the overall performance is slightly better on the USYD Campus dataset that in the Kitti dataset. This is expectable as the 3DCNN was trained on data provided by the first mentioned dataset. Nonetheless, the accuracy is still high in both data setups.

In the light of the experiments, it can also be conclude that YOLO v3 is the best object detector to use in the proposed pipeline as it provides precise and sharp AOIs. Thus, the 3D CNN was fed with accurate tridimensional object data. This fact allowed it to perform accurately in both data setups.

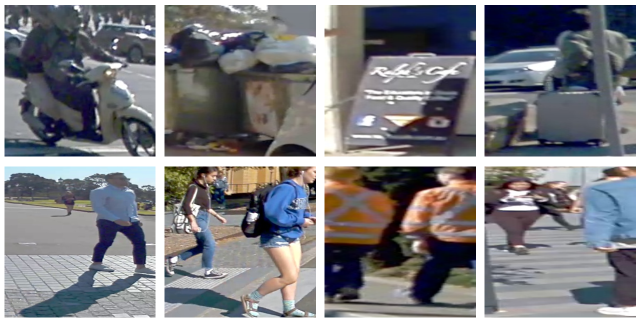

Figure 4: The top row depicts samples rejected as pedestrian by the 3D CNN. The bottom row shows samples incorrectly confirmed as pedestrians. Note that all these samples were detected as pedestrians by the 2D object detectors.

Tiny-YOLO provided the worst results when used in the pipeline as it is the weakest approach to object detection. It usually provides poorly fitted AOIs. It is common to find objects which were not completely inside the AOI, and sometimes the same AOI depicted two pedestrian if they were close enough. As a result, the 3D CNN is fed with erroneous and its accuracy decreased.

The pipeline involving MobileNetSSD as the object detector performed slightly better than YOLO v3 on the USYD campus dataset experiment, and the contrary on the Kitti dataset. This difference in the accuracy rates could be due to the features of the images. The first mentioned dataset provides 16:10 aspect ratio images whilst the images of the Kitti dataset are a panoramic reconstruction much more wide than tall. This fact is likely exceeding the multi-scale features of this network, thus harming its accuracy.

Figure 4 shows some samples correctly rejected as no pedestrians and incorrectly confirmed as pedestrians. In these cases it can be seen that the AOIs are poorly fitted.

It is worth noting that the 3D CNN provided accurate predictions in each case, as the results show. Nonetheless, its performance is highly dependent on the sharpness of the provided data. Thus, it greatly relies of the accuracy of the object detector. In addition, it can be conclude that the 3D CNN is also highly agnostic to the resolution of the 3D data. Note that the USYD campus dataset provide 16 beams laser data and Kitti features 64 beams laser data. This is likely due to the binary voxelization procedure. The resolution of the 3D data is also related to the distance limitation. For instance, for the USYD campus dataset the limit is 10m, but there were no distance limit set for the Kitti experiment.

Finally, the end-to-end pipeline run at between 7 and 10 fps, including data grabbing and communication overhead. The variance is related to the object detector and the number of detected objects. A feedforward step of the 3D CNN only takes 52ms. The slowest part of the pipeline regards to the lidar-to-image data projection and the voxelization process of each sample. The measures are computed by running the method in the hardware described in section IV and averaged across the experiments.

VI. Future Work

In order to improve the generalization capabilities of the 3D CNN, we plan to involve different datasets and data augmentation techniques. Thus, the overall accuracy is expected to improve. We also plan to involve the object detection stage and the 3D CNN in the same architecture so the 2D and 3D features are jointly learned and the runtime reduced.

We also plan to further test the relationship between the laser resolution and the suitably of the voxelized representation of the 3D data so we can properly set the limitations in this regard.

References

[1] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1912–1920.

[2] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," *CoRR*, vol. abs/1604.03265, 2016. [Online]. Available: http://arxiv.org/abs/1604.03265

[3] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. USA: IEEE Computer Society, 2015, p. 945–953. [Online]. Available: https://doi.org/10.1109/ICCV.2015.114

[4] A. Kanezaki, "Rotationnet: Learning object classification using unsupervised viewpoint estimation," *CoRR*, vol. abs/1603.06208, 2016. [Online]. Available: http://arxiv.org/abs/1603.06208

[5] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," *CoRR*, vol. abs/1605.08359, 2016. [Online]. Available: http://arxiv.org/abs/1605.08359

[6] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2511–2519.

[7] B. Shi, S. Bai, Z. Zhou, and X. Bai, "Deeppano: Deep panoramic representation for 3-d shape recognition," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2339–2343, Dec 2015.

[8] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval," in *Eurographics Workshop on 3D Object Retrieval*, I. Pratikakis, F. Dupont, and M. Ovsjanikov, Eds. The Eurographics Association, 2017.

[9] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval," *Computers & Graphics*, vol. 71, pp. 208 – 218, 2018.

[10] A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez, "Pointnet: A 3d convolutional neural network for real-time object class recognition," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 1578–1584.

[11] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 922–928.

[12] C. Ma, W. An, Y. Lei, and Y. Guo, "Bv-cnns: Binary volumetric convolutional networks for 3d object recognition," in *BMVC*, 2017.

[13] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," *arXiv preprint arXiv:1803.04249*, 2018.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016. [Online]. Available: http://arxiv.org/abs/1612.00593

[15] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks," *CoRR*, vol. abs/1711.08241, 2017. [Online]. Available: http://arxiv.org/abs/1711.08241

[16] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. de la Escalera, "Birdnet: a 3d object detection framework from lidar information," *CoRR*, vol. abs/1805.01195, 2018. [Online]. Available: http://arxiv.org/abs/1805.01195

[17] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *CoRR*, vol. abs/1711.06396, 2017. [Online]. Available: http://arxiv.org/abs/1711.06396

[18] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," *CoRR*, vol. abs/1903.08701, 2019. [Online]. Available: http://arxiv.org/abs/1903.08701

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[22] J. Pedoeem and R. Huang, "YOLO-LITE: A real-time object detection algorithm optimized for non-gpu computers," *CoRR*, vol. abs/1811.05588, 2018. [Online]. Available: http://arxiv.org/abs/1811.05588

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," 2015, cite arxiv:1512.02325Comment: ECCV 2016. [Online]. Available: http://arxiv.org/abs/1512.02325

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[25] A. Garcia-Garcia, F. Gomez-Donoso, J. G. Rodríguez, S. Orts-Escolano, M. Cazorla, and J. A. López, "Pointnet: A 3d convolutional neural network for real-time object class recognition," in *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, 2016, pp. 1578–1584. [Online]. Available: https://doi.org/10.1109/IJCNN.2016.7727386

[26] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning," *arXiv e-prints*, p. arXiv:1908.02983, Aug 2019.

[27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.