Human Detection in the Depth Map Created from Point Cloud Data

Adam Ligocki[1 [0000−0002−6813−4318]] and Ludek Zalud[1 [0000−0003−2993−7772]]

Brno University of Technology,
Faculty of Electrical Engineering and Communication,
Brno, Technicka 12, Czechia
adam.ligocki@vutbr.cz

Abstract. This paper deals with human detection in the LiDAR data using the YOLO object detection neural network architecture. RGB-based object detection is the most studied topic in the field of neural networks and autonomous agents. However, these models are very sensitive to even minor changes in the weather or light conditions if the training data do not cover these situations. This paper proposes to use the LiDAR data as a redundant, and more condition invariant source of object detections around the autonomous agent. We used the publically available real-traffic dataset that simultaneously captures data from RGB camera and 3D LiDAR sensors during the clear-sky day and rainy night time and we aggregate the LiDAR data for a short period to increase the density of the point cloud. Later we projected these point cloud by several projection models, like pinhole camera model, cylindrical projection, and bird-view projection, into the 2D image frame, and we annotated all the images. As the main experiment, we trained the several YOLOv5 neural networks on the data captured during the day and validate the models on the mixed day and night data to study the robustness and information gain during the condition changes of the input data. The results show that the LiDAR-based models provide significantly better performance during the changed weather conditions than the RGB-based models.

Keywords: LiDAR data · RGB camera · Point Cloud · projection · YOLO · Object Detection · Neural Network · DCNN.

1 Introduction

These days, we can see the growing number of various applications of autonomous agents in Advanced Driving Assistant Systems (ADAS) and in many fields of industrial automation or even in our homes. These systems have very high demands on the security of the people that interact with them. For example, in autonomous cars, we talk about vehicles' crew that is permanently at risk as the cars are moving with very high velocities or the very poorly protected pedestrians in the moment of collision with the vehicle. On the other hand, in the industry, we see various systems and robotic manipulators that use large forces to handle very heavy cargoes. All these systems have to be designed with the security systems in mind, that would prevent them from harming living beings, even if they break the security rules.

Many different approaches allow detecting humans' presence and make it possible to avoid the collision. For autonomous cars, the most common is the usage of RGB cameras, LiDARs, radars, etc. In the industry sector, there are cameras, mechanical and optical branches in use. Overall, if we want to detect the presence of a human in some area, RGB cameras combined with the neural network object detectors are these days the easiest and the most reliable method to do so.

Fig. 1. During the night, the RGB camera was covered by the water from rain and partially blinded. It significantly affects the inference of the neural network object detectors. At the very same time, the LiDAR sensor worked without any significant degradation of the data quality.

In many edge cases, the RGB camera is not a source of reliable data. Situations, like light condition changes, bad weather, rain, fog, etc., make the RGB camera nearly useless (see the Figure 1). Therefore, we decided to study the LiDAR sensor and its usage in detecting humans as the laser time of flight (TOF) sensor provides significantly higher robustness regarding unfavorable conditions. In the end, the LiDAR-based human detector could help handle edge situations when the common RGB-based detection methods fail.

2 Related Works

The primary motivation for our work is that LiDAR sensors are more robust during light or weather conditions changes than RGB cameras. For example, the standard camera optics are blinded even during the soft rain when the LiDAR sensors can handle these conditions up to rain intensity of several mm per hour [1], [2], and [3].

The field of object detection in LiDAR data is quite well covered, not only by classic point cloud processing method, like clustering [4], [5], which could also be applied on pedestrian detection [6], [7]. On the other hand, these days the application of the neural network is the most frequently used approach. Let us mention the well known PointNet++ [8], Complex-Yolo [9], or the BirdNet [10].

All the papers mentioned above deal with the point cloud data. We handle the problem differently in this work, and we process point cloud data like an image. We based it, on the idea of transfer learning [11]. Neural networks require an enormous amount of

training data, which is very difficult to collect. However, we can use the model trained on a similar problem, finetune it, and adjust it to the problem we want it to deal with.

Our idea is to use the pre-trained RGB neural network and apply it on depth map data generated from the LiDAR point cloud data. In this way, we can cover not only human detection in the field of autonomous cars but also in the indoor and other security applications, where the RGB-Depth cameras are used [12], [13].

Object detection in depth maps images using neural networks is not a new idea. There are many papers, like [14], [15] or [16], but they all focus on a fusion of the depth map and common RGB camera data. Only very few papers study the raw depth map and compare the different projection types and points of view [17], but non of those dataset compares the RGB and point cloud data in the way, we do.

3 Dataset

To train and evaluate all the proposed models, we created our own dataset based on the open source Brno Urban Dataset (BUD) [18]. The Brno Urban Dataset is a publically available set of real-life road traffic records that contains data from four RGB cameras, a single thermal camera, three 3D LiDAR sensors, IMU, and an RTK GNSS receiver. The dataset also provides the calibration data for the physical layout of the sensors as homogeneous transformations w.r.t. the IMU in the center of the sensory framework and internal calibration parameters for all cameras.

3.1 Depth Image Generation

We used four recording sessions from the BUD in total. It comprises of about half an hour of the data recorded during both day and night time and we processed it using the open-source Atlas Fusion framework [19]. It allows us to aggregate point cloud data from the Livox Horizon LiDAR sensor, pair them with the corresponding image from the very front RGB camera, and project this point cloud data into the RGB camera plane. For projecting point clouds into the camera frame, we used the pinhole camera model. All the generated images are 1920x1200px, and we projected each point as a 11x11px square. This way, we created the depth map-like image for every single RGB frame. The grayscale intensity of the projected point was estimated based on the point's distance from the camera's optical center as linear regression, 255-pixel value for zero distance, and the 0 value for points in the distance of 50m, or more.

In total, we created about 20 000 RGB-depth map pairs. For those, we extract only the pairs that contained pedestrians in the RGB image. Also, as the cameras capture images at 10 Hz, we removed the following nine pairs after every single selected RGB-depth pair containing pedestrians. This way, we removed from the dataset the RGB-depth map pairs that cover the same scene, only captured from a slightly different perspective.

In total, we selected four hundred pairs of RGB images and corresponding depth maps. All containing at least one pedestrian.

Fig. 2. The example of the RGB images (left) with the corresponding depth map image (right) generated by projecting point cloud to camera frame, using the pinhole camera model.

3.2 Different Projections

As a part of the paper, we studied not only the simple pinhole camera projections to the physical camera, but we also tested the other types of projection of the point clouds to compare the possible information gain in the pedestrian detection task.

First we proposed creating the depth maps for a virtual camera that would take place three meters above the physical one, and the virtual camera would be tiled by 20 degrees below the horizon. Then, we tried to validate the hypothesis that the neural network would deal better if it would combine the visual information of the pedestrian's presence with the shadow in the point cloud data behind him. The depth map images for the virtual camera have the same parameters as the depth map created by projecting point cloud to the physical camera, 1920x1200px, and points were projected as 11x11px squares. We set the pixel intensity in the same way as the projection to the physical camera.

Next, we used the spherical projection into the physical camera's optics center. We created the 1800x600px blank image, and for each point in the 90x30 deg field of view, we projected it into the image as a 11x11px square. This way, each row and column in the image corresponds to 0.05 deg in the spherical coordinates. Finally, the pixel intensity was estimated in the same way as the previous two projection methods.

The very last projection type is the bird view projection, where we took the 50x30m area in front of the camera, where each pixel represented the 0.1x0.1 cell on the ground. This results in a 500x300px image. For every point in the point cloud, we used its X and Y position coordinates to estimate the corresponding image pixel, and we set this

pixel to the value given by the linear regression between the -2m (0) and the 3m (255) of the Z coordinate of the point. If several points occupied the same cell, we projected the highest one.

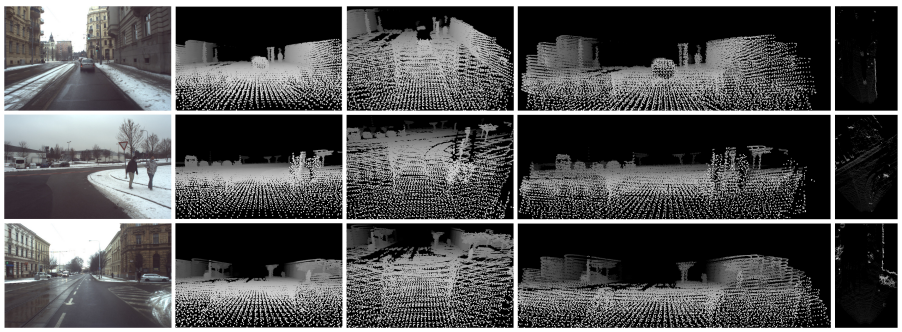

Fig. 3. Visualization of different projection types of the same scenes. From left to right: real RGB, pinhole projection to physical camera, pinhole projection to virtual camera, spherical projection to physical camera and bird-view projection.

This way, we created four different depth maps by projecting the aggregated point cloud for each captured RGB image.

3.3 Annotation

We selected the dataset, which contains 400 RGB images, 300 captured during the day and 100 images captured during the rainy night. For each RGB image, we generated four different projections of the point clouds from the corresponding time. In total, it gives us 2000 images. We annotated them all by hand using the LabelImg tool `https://github.com/tzutalin/labelImg`.

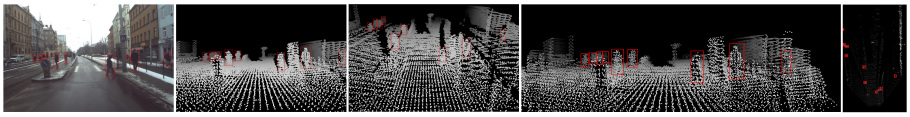

Fig. 4. Example of annotations for all used types of projection. The layout of different projections is same, like in Figure 3.

3.4 Morphological Modifications

We aggregate the captured point cloud data for one second. Still, the images created by projecting the point cloud data to the camera frame are pretty sparse. Thus we tried to

extend the area of each point by applying the morphological operation of dilatation on each depth map image. We applied the dilatation using the kernels of 3x3, 5x5, 7x7, and 9x9px size.

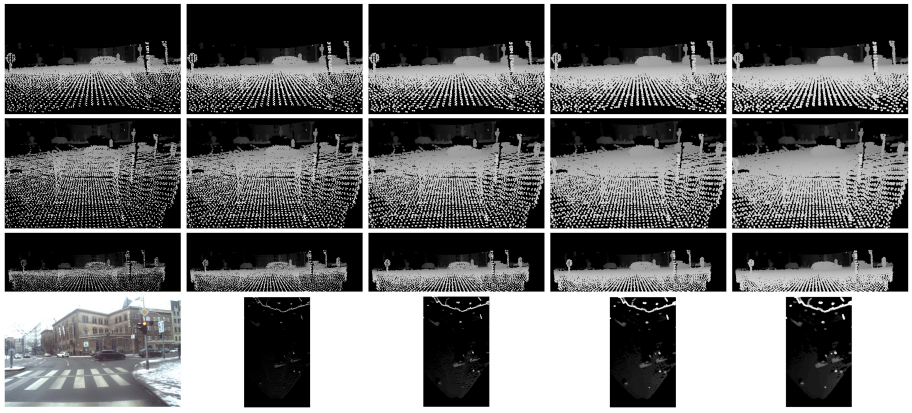

Fig. 5. Visualization of different projections (in rows) and the application of morphological operation of dilatation. First image in every column is always the original image, followed by the dilatation by kernel of size 3, 5, 7, 9. In last row the RGB image is appended. For bird-view projection only kernels sizes of 3, 5 and 7 were applied.

Annotations for images modified by the dilatation stay the same as the annotations for the original image, which serves as an input to the morphological operation.

3.5 Summary

In total, we made 400 RGB images, 300 captured in the day, 100 captured in the night, during the rain. We generated the depth map for each RGB image by projecting point cloud to the physical camera, to the virtual camera placed 3m above the physical camera, the spherical projection to the real camera coordinates, and the bird view projections. Additionally for each physical camera projection, virtual camera projection and spherical projection image, we extended the dataset by applying the morphological operation of dilatation with a kernel size of 3, 5, 7, and 9. For the bird view projection, we used only kernels of sizes 3, 5, and 7.

Summing it all together, we prepared 8000 different images split into the 20 small training datasets, each of 400 images.

4 Models Training and Evaluation

To train the neural network models, we used the YOLOv5 framework [20]. It implements the entire YOLO architecture, based on the YOLOv3 [21] for four models of

different complexities, S, M, L, and X. The framework also allows using training data augmentation, which allows models to reach better results even for small datasets.

The YOLO architecture is an end-to-end neural network model that takes the raw image on the input and proposes a tensor representing the set of proposed bounding boxes on the output. The output tensor is a 3D data structure, where the X and Y coordinates correspond to the grid cell that divides the input image. The depth dimension of the tensor encapsulates the parameters of the proposed bounding boxes. See the Figure 6.

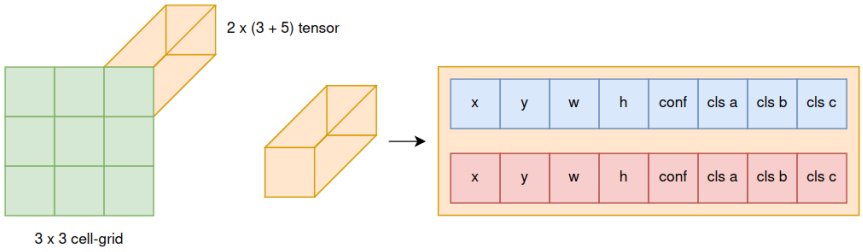

Fig. 6. The YOLO [22] is the full end-to-end neural network. On the input, it takes raw N-channel images and split them to the SxS cell grid. On the output of the neural network, there is a tensor of SxSxM size. The M represents the length of the vector that holds the numbers that represent parameters of proposed bounding boxes, like position, dimension, class score, and bounding box confidence.

To evaluate the performance of the trained models, we used the metrics commonly used in the object detection tasks, the mAP(0.5) and mAP(0.5:0.95). Additionally, we also added the F1 score, which expresses the relation between the recall and the precision of the model.

Our entire dataset is divided into 20 smaller subsets - one RGB subset, five subsets of depth maps created by projecting point clouds by pinhole camera model into the camera frame (one original subset and four derived subsets generated using the morphological dilatation), five subsets of depth map by projecting point clouds to the virtual camera above the physical one, five subsets for spherical projection, and four subsets of bird view projections.

Each subset comprises 400 images, 300 captured in the day and 100 captured during the night. Thus, we used 240 day-time images to train the neural network model, 60 day-time images to validate the model during the training process, and the 100-night images we use to estimate the robustness of the model as the light conditions change.

We trained three YOLOv5 X models for every one of 20 subsets. We used the transfer learning technique, using the pre-trained YOLOv5 weights trained on the RGB data and trained to detect bounding boxes of 80 COCO classes [23]. To adjust the neural network for our task, we modified the last layer of the model so it detects only one class on the output, the pedestrians. After training three models on each subset, we evaluated each of them and measured their performance on the day-time validation data using the

mAP(0.5), mAP(0.5:0.95), the F1 score metrics and calculated the average for each of those criteria. See the results in the Table 1.

Table 1. The table shows the performance of trained neural networks, measured on validation data captured during the day. The number "Kernel Size" gives the NxN dimensions of the kernel used to dilate the depth map before the training process.

Daytime Dataset	Dil. Kern. Size	Precision	Recall	mAP(0.5)	mAP(0.5:0.95)	F1
RGB	-	0.674	0.843	0.830	0.457	0.749
Pinhole Projection (Physical Cam)	-	0.514	0.672	0.635	0.339	0.583
	3	0.562	0.707	0.667	0.360	0.627
	5	0.604	0.691	**0.687**	**0.363**	0.644
	7	0.606	0.674	0.656	0.362	0.638
	9	0.614	0.687	0.664	0.356	0.648
Pinhole Projection (Virtual Cam)	-	0.548	0.645	0.616	0.341	0.592
	3	0.594	0.657	0.644	0.367	0.624
	5	0.602	0.696	**0.681**	**0.391**	0.645
	7	0.583	0.660	0.647	0.368	0.645
	9	0.576	0.669	0.656	0.385	0.619
Spherical projection (Physical Cam)	-	0.524	0.647	0.607	0.282	0.579
	3	0.521	0.657	0.605	0.284	0.581
	5	0.618	0.673	**0.622**	0.299	0.645
	7	0.573	0.646	0.615	**0.306**	0.608
	9	0.474	0.701	0.616	0.305	0.566
Bird View	-	0.482	0.775	**0.694**	0.291	0.595
	3	0.504	0.718	0.673	**0.307**	0.592
	5	0.456	0.699	0.631	0.297	0.552
	7	0.431	0.716	0.637	0.295	0.538

Later, for each projection type we selected the models trained on the morphologically modified dataset with the best average mAP(0.5) score above others. We assume that the selected models are trained on those datasets, where the morphological dilatation upscaled projected points to the size, meaning that the neural network can detect the objects with the best performance. We later evaluated these selected models on the night-part of the same projection type and morphological modification. The results are shown in the Table 2 below.

5 Discussion

The results of the models evaluation are shown in the Table 1 (day data) and in the Table 2 (night data).

We trained three neural networks on a day data for each subset and calculated the average over their results. For each projection type, we chose the best average results

Table 2. Performance of trained models on the data captured during the rainy night. Compared to the "Day Data," LiDAR data-focused models outperform the RGB object detector during the night.

Night Dataset	Dil. Kern. Size	Precision	Recall	mAP(0.5)	mAP(0.5:0.95)	F1
RGB	-	0.643	0.580	0.582	0.259	0.609
Pinhole Projection (Physical Cam)	5	0.689	0.900	**0.879**	**0.482**	0.780
Pinhole Projection (Virtual Cam)	5	0.712	0.872	0.871	0.420	**0.784**
Spherical projection (Physical Cam)	5	0.699	0.856	0.814	0.300	0.770
Bird View	-	0.475	0.816	0.716	0.270	0.600

concerning the kernel size. These results are shown in Table 1. Here the performance on the RGB data is mentioned as a reference to compare object detection in the depth map with the RGB-based state-of-the-art. The depth map-based approaches give worse results compared to the RGB. However, if we compare only the models trained on the depth map subsets, the best results are reached by the simple pinhole camera projection into the physical camera, placed on the roof of the car, and by the same projection to the virtual camera that is placed three meters above the physical one. On the other hand, the spherical projection and bird-view projection did not show any advantage, and their performance is significantly worse.

Considering the kernel size, Table 1 shows that in most cases, we get the best results for the dilatation with a kernel size of 5x5. If we consider that the point cloud data were projected to the image frame as 11x11 squares, applying the 5x5 dilatation, we can assume that for the 1920x1200 image, the best results were reached using the 19x19px size of projected points. For bird-view projection, the dilatation did not help to get any better performance. It corresponds with the smaller size of the bird-view image and the smaller size of the objects in the image. However, this paper studies the different sizes of the projected points very briefly. In the future, this topic could be extended.

Later, we used these best models selected by the performance on the day data, and we evaluated them on the corresponding data that we recorded during the night. The results are presented in Table 2. Here we see, that all the depth-based models outperform the RGB object detector.

Figure 7 visualizes several detections for each projection type.

6 Conclusion

This paper proposes an experiment that evaluates different projection methods of LiDAR's point cloud data into the 2D depth map images and tests the YOLO neural network's capability to detect objects in these images. In total, we created 20 different 400-images datasets and trained the YOLOv5 X neural network model on the data cap-

tured during the day. Later we select the best model for each projection type, and we evaluate them on the data recorded during the rain and night.

The results confirmed our original hypothesis that object detection on depth map images shows significantly worse results during good weather conditions (0.830 for RGB vs. 0.687 mAP(0.5) for depth map), but as the light and weather worsen, the LiDAR-based depth map object detection outperforms the object detection on RGB data (0.582 for RGB vs. 0.879 mAP(0.5)).

Acknowledgement

The work has been performed in the project ArchitectECA2030: Trustable architectures with acceptable residual risk for the electric, connected and automated cars, under grant agreement No 877539/8A20002. The work was co-funded by grants of Ministry of Education, Youth and Sports of the Czech Republic and Electronic Component Systems for European Leadership Joint Undertaking (ECSEL JU). The work was supported by the infrastructure of RICAIP that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857306 and from Ministry of Education, Youth and Sports under OP RDE grant agreement No CZ.02.1.01/0.0/0.0/17_043/0010085.

References

1. C. Goodin, D. Carruth, M. Doude, and C. Hudson, "Predicting the influence of rain on lidar in adas," *Electronics*, vol. 8, no. 1, p. 89, 2019.
2. A. Filgueira, H. González-Jorge, S. Lagüela, L. Díaz-Vilariño, and P. Arias, "Quantifying the influence of rain in lidar performance," *Measurement*, vol. 95, pp. 143–148, 2017.
3. M. Kutila, P. Pyykönen, H. Holzhüter, M. Colomb, and P. Duthon, "Automotive lidar performance verification in fog and rain," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1695–1701.
4. A. Börcs, B. Nagy, and C. Benedek, "Instant object detection in lidar point clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 7, pp. 992–996, 2017.
5. M. Sualeh and G.-W. Kim, "Dynamic multi-lidar based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, p. 1474, 2019.
6. K. Liu, W. Wang, and J. Wang, "Pedestrian detection with lidar point clouds based on single template matching," *Electronics*, vol. 8, no. 7, p. 780, 2019.
7. H. Wang, B. Wang, B. Liu, X. Meng, and G. Yang, "Pedestrian recognition and tracking using 3d lidar for autonomous vehicle," *Robotics and Autonomous Systems*, vol. 88, pp. 71–78, 2017.
8. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.
9. M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.
10. J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "Birdnet: a 3d object detection framework from lidar information," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3517–3523.

11. C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.
12. K. Zhou, A. Paiement, and M. Mirmehdi, "Detecting humans in rgb-d data with cnns," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*. IEEE, 2017, pp. 306–309.
13. P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, "Rgb-d-based human motion recognition with deep learning: A survey," *Computer Vision and Image Understanding*, vol. 171, pp. 118–139, 2018.
14. M. J. Seikavandi, K. Nasrollahi, and T. B. Moeslund, "Deep car detection by fusing grayscale image and weighted upsampled lidar depth," in *Thirteenth International Conference on Machine Vision*, vol. 11605. International Society for Optics and Photonics, 2021, p. 1160524.
15. Z. Guo, W. Liao, Y. Xiao, P. Veelaert, and W. Philips, "Deep learning fusion of rgb and depth images for pedestrian detection," in *30th British Machine Vision Conference*, 2019, pp. 1–13.
16. X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.
17. Y. Luo, C. Zhang, M. Zhao, H. Zhou, and J. Sun, "Where, what, whether: Multi-modal learning meets pedestrian detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 065–14 073.
18. A. Ligocki, A. Jelinek, and L. Zalud, "Brno urban dataset-the new data for self-driving agents and mapping tasks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3284–3290.
19. ——, "Atlas fusion–modern framework for autonomous agent sensor data fusion," *arXiv preprint arXiv:2010.11991*, 2020.
20. G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, A. Hogan, lorenzomammana, yxNONG, AlexWang1900, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, F. Ingham, Frederik, Guilhen, Hatovix, J. Poznanski, J. Fang, L. Yu, changyu98, M. Wang, N. Gupta, O. Akhtar, PetrDvoracek, and P. Rai, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370
21. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
22. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
23. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

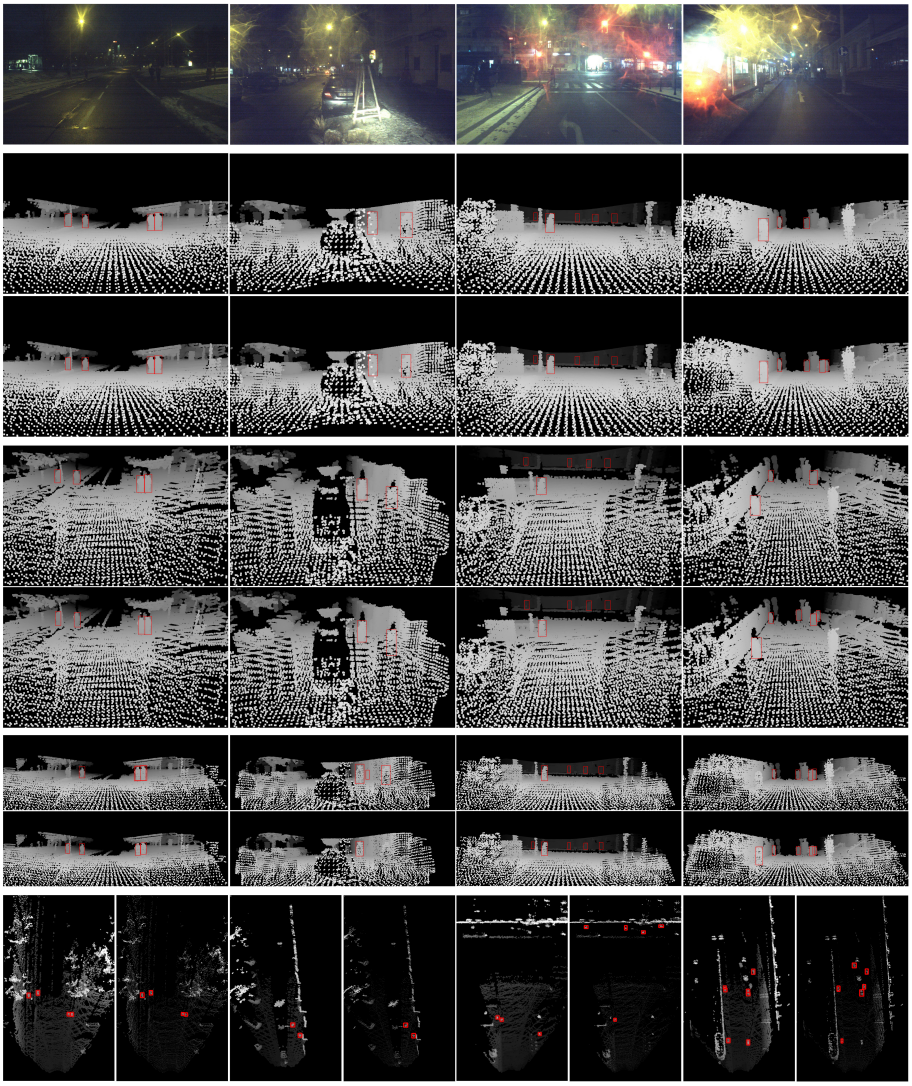

Fig. 7. Each column shows the data captured at the exact same moment. On the top, the RGB image illustrates the environment in front of the car. Then follow by the row the simple pinhole projection to the physical camera, pinhole projection to the virtual camera, spherical projection to the physical camera, and bird view projection. Always the image with neural network's detections is above the ground truth annotation. For bird-view projection, the image with detections is on the left and ground truth on the right side of the column.