SEMANTIC SCENE UNDERSTANDING FOR THE AUTONOMOUS PLATFORM

B. Vishnyakov *, Y. Blokhinov, I. Sgibnev , V. Sheverdin, A. Sorokin, A. Nikanorov, P. Masalov, K. Kazakhmedov, S. Brianskiy, E. Andrienko, Y. Vizilter

FGUP «State Research Institute of Aviation Systems», Russia, 125319, Moscow, Viktorenko street, 7 - (vishnyakov, yuri.blokhinov, sgibnev, sheverdin, ans, avnikanorov, masalov, kkirill, sbrianskiy, viz)@gosniias.ru

KEY WORDS: multi-sensor platform, autonomous vehicle, SLAM, CNN, dynamic scene analysis, semantic segmentation, off-road, autonomous driving, camera calibration, LiDAR calibration.

ABSTRACT:

In this paper we describe a new multi-sensor platform for data collection and algorithm testing. We propose a couple of methods for solution of semantic scene understanding problem for land autonomous vehicles. We describe our approaches for automatic camera and LiDAR calibration; three-dimensional scene reconstruction and odometry calculation; semantic segmentation that provides obstacle recognition and underlying surface classification; object detection; point cloud segmentation. Also, we describe our virtual simulation complex based on Unreal Engine, that can be used for both data collection and algorithm testing. We collected a large database of field and virtual data: more than 1,000,000 real images with corresponding LiDAR data and more than 3,500,000 simulated images with corresponding LiDAR data. All proposed methods were implemented and tested on our autonomous platform; accuracy estimates were obtained on the collected database.

1. INTRODUCTION

The autonomous car market is currently growing at an existential rate and many companies develop their own concepts of driverless vehicles. A self-driving car, also called an autonomous vehicle, is a vehicle that uses a combination of sensors, cameras, radars and artificial intelligence, to travel between destinations without the need of any human effort.

Scientific community publishes huge number of papers on the topics of object detection, scene segmentation, 3D-reconstruction using cameras and LiDARs, radars. These algorithms combination allows us to develop high level algorithms of autonomous driving. However, most of the driving algorithms are based on the vector map of the roads. So, autonomous driving in off-road conditions, in the countryside is still a challenging problem. The solution requires robust algorithms of semantic segmentation, three-dimensional scene reconstruction, object detection. All these algorithms work much better in the cities than in the countryside.

In this paper we describe our multi-sensor off-road platform for data collection and algorithm testing. We propose a new, fully automatic technique for mutual calibration of machine vision cameras and LiDARs, discuss algorithms for real-time semantic 3D-scene reconstruction.

2. AUTONOMOUS PLATFORM

Autonomous platform is a relatively large vehicle with dimensions close to real cars (1.8m wide, 4.4m long).

2.1 Sensors

The core of the autonomous platform is a computer vision hardware complex, which consists of ten short focus (5mm lens) and two long focus (25mm lens) Prosilica GT2050C machine vision cameras, four SWIR cameras Goldeye G-032 SWIR TEC1, four Velodyne VLP-16 LiDARs. This system allows us to collect video and three-dimensional data and try out algorithms for three-dimensional reconstruction, semantic segmentation and obstacle classification. This vision system is mounted on a metal frame support that allows one to change the distance between the cameras and quickly install or remove other sensors if necessary. In addition, two AXIS M5525 PTZ cameras for object detection are places on the platform. This computer vision subsystem is designed to collect data and try out algorithms for object detection and recognition, semantic segmentation, three-dimensional scene reconstruction.

The sensors location on the platform is shown in Figure 1.

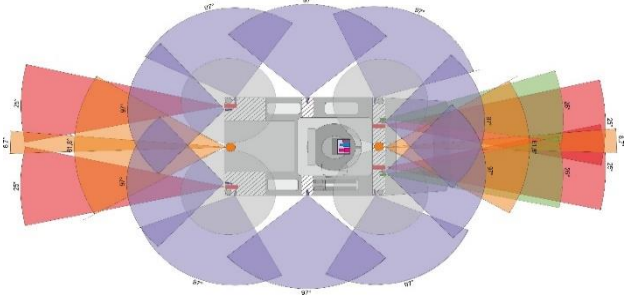


Figure 1. Ten short focus cameras (purple), two long focus cameras (green), four LiDARs (grey circles), two PTZ cameras (orange), four SWIR cameras (light red)

Hardware processing part consist of seven computing units – Vecow RCS-9430FHR-RTX2080-256 industrial computer based on Intel Core i7-7700 processor, Nvidia GeForce RTX 2080 graphics card and a special four-channel gigabit network card with power over the network (PoE) function – PE-1004. All machine vision cameras are connected to the PE-1004 board since each camera generates a data stream of approximately 1 Gbit per second. Other devices are connected to a gigabit switch and are in the same local area network. Also, a Delphi ESR-2.5 radar, GPS-receiver and xsens inertial system can be mounted on the vehicle.

* Corresponding author

We use UPS APC Smart-UPS SRT 1000VA / 900W uninterruptible power supply with five 1500VA batteries, which allows the computing unit and the set of sensors of the vision system to run continuously up to 8 hours.

2.2 Software

We developed special software using ROS2 platform on Ubuntu 18 basis, which allows us to synchronously record data from all sensors in the system, including cameras, LiDARs, radar, GPS-receiver and inertial system, to a specialized storage called rosbag. Data streams from cameras and LiDARs are synchronized at a hardware level with synchronization cables and over PTP/PPS protocols.

An optional remote Wi-Fi connection of the operator to the computing unit is also optionally provided for the purpose of monitoring data collection processes or testing computer vision algorithms.

3. VIRTUAL SIMULATION

A lot of scientific labs and groups of engineers use virtual simulation as a most affordable way to generate extra data for training of neural networks. We also use virtual simulation to get image and LiDAR data in different conditions.

We chose Unreal Engine, a game engine developed and supported by Epic Games, as a basic simulation tool. A game engine (not a professional one, for example, Vega Prime) was chosen due to the fact that the game engines currently provide the most realistic scene visualization. Since 1998 (when the first version of the Unreal engine was released), various versions of the engine have been used in more than a hundred games and a thousand of other projects, including scientific projects and virtual simulation tools.

3.1 Modeling process

Our modeling process consists of three parts. The first part is object placement and setting the routes for dynamic objects such as people, cars, etc. The second part is the virtual travel along the desired route. The third part is setting the weather conditions and recording of images and LiDAR data. When recording video from the camera, the calculation and recording of the boundaries of objects on the screen are also performed. Borders are calculated using Ray tracing.

The virtual world was created using the landscape from the tech demo “A Boy and His Kite” with a total area of about 10 km2 with maximum detail on an area of about 2.5 km^2, as well as our own off-road scene (about 4 km^2). Also, we created some extra 3D models, such as people, cars, stones, trees, grass and particle filters to simulate weather conditions and increase the overall quality of the simulation (Figures 2 and 3).

Figure 2. First virtual scene sample

Figure 3. Second virtual scene sample

3.2 LiDAR modelling

When modeling VLP-16 LiDAR, single measurement 16 rays are emitted at different polar angles from the point where the LiDAR is mounted, and for each ray the distance to the nearest object that this ray intersects and information about this object, is written to a file.

The saved data can be visualized using standard VeloView application by Velodyne. You can see the example of LiDAR data for simulated travels in Figure 4, where the objects of interest (for example, a person, a tree, a stone are highlighted in yellow).

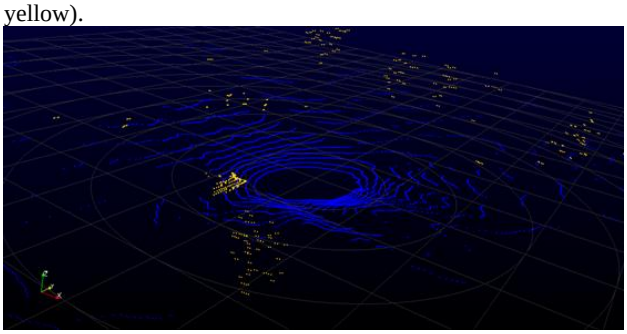


Figure 4. Modelled LiDAR data

3.3 Automatic image and LiDAR data annotation

We developed automatic annotation modules for both object detection and segmentation tasks. Our dataset consists of more than 3,500,000 images and corresponding LiDAR data with automatic annotation of all objects (2D bounding boxes of objects and obstacles in images) and about 10% of images having segmentation masks.

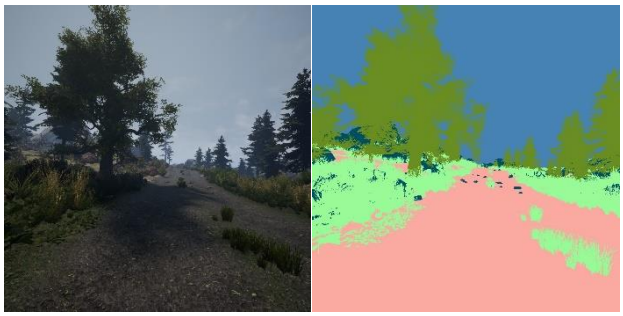


Figure 5. Original image and its segmentation mask

4. AUTOMATIC CAMERA AND LIDAR CALIBRATION

There is a number of works devoted to the calibration process of video cameras with LiDAR: (Pusztai, Hajder, 2017), (Park et al., 2014), (Pereira et al., 2016), (Xu, Li, 2014), (Guindel et al., 2017), (Chai et al., 2018), (Dhall et al, 2017). All of them could be classified into two groups, according to the sensor orientation method each algorithm is based on. Algorithms from the first group use 2D and 3D matching, in other words, coordinates of some points that were directly obtained from cameras and LiDARs (Pusztai, Hajder, 2017), (Park et al., 2014), (Pereira et al., 2016), (Xu, Li, 2014).

Algorithms from the second group are based on the 3D matching for orientation calculation (Guindel et al., 2017), (Chai et al., 2018), (Dhall et al, 2017). Thus, a cameras stereoscopic pair is essential for 3D coordinates calculation of angles on a pair of images. Alternatively, ARUCO markers (Garrido-Jurado et al., 2014), (Romero-Ramirez et al, 2018) could be used for 3D coordinates calculation.

However, a major part of mentioned papers requires either a human operator inference or sophisticated calibration facilities usage for particular points detection (Pereira et al., 2016), (Guindel et al., 2017), (Chai et al., 2018).

Current work argues automatic calibration process using different types of sensors combined with simplest calibration equipment.

4.1 Calibrated camera and LiDAR system

The system we calibrate consists of two, three or four Prosilica GT2050C cameras with 5mm and 25mm lenses and a one or two Velodyne VLP-16 LiDARs.

Both the camera and LiDAR system that needs to be calibrated are shown in Figure 6. The system consists of two Prosilica GT2050C cameras with 5mm lenses forming a stereo pair, two Prosilica GT2050C cameras with 25mm lenses forming a stereo pair, and a Velodyne VLP-16 LiDAR.

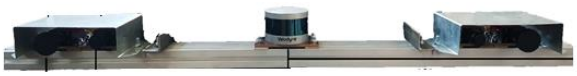


Figure 6. Camera and LiDAR system with for cameras and one LiDAR

4.2 Cameras and calibration board

We use 1m x 1m calibration board with 25 ArUco markers on it. Calibration board is bolted to the revolving plate, that is fixed on the base. Revolving plate allows us to change angles positions without shifting the whole frame. ARUCO markers provide simple and accurate angles detection. We attached light-reflecting stripes to the right edge of the calibration board, so we can easily detect its edges in LiDARs point cloud (Figure 7).

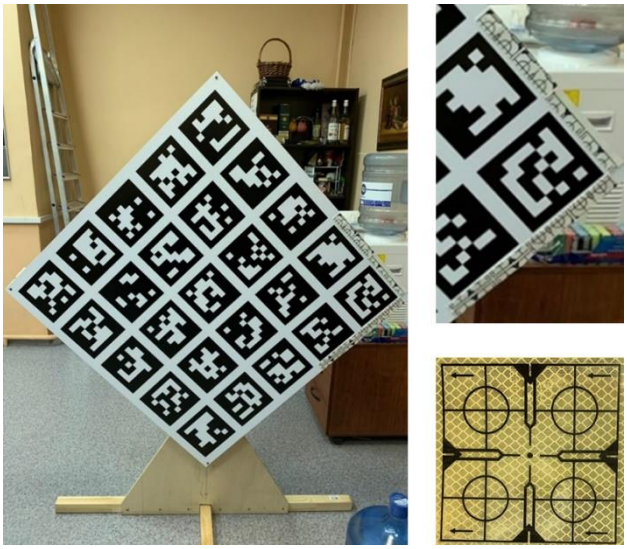


Figure 7. Calibration board. Light-reflection stripes are attached to the right edge of the board.

The accuracy of the developed procedures is estimated by reprojection error. The projection matrix is calculated using intrinsic parameters of the camera K, the rotation matrix R and the displacement vector t as follows:

$$P = KRt.$$

4.3 LiDAR and camera calibration

Finding camera and LiDAR mutual position requires two sets of points: calibrations plate angle position, calculated in 3D coordinates and respective edge positions on both images. Whole calibration procedure can be divided into two stages: data preparation and transformation parameters calculation. First stage requires collecting a certain number of calibration plate images from cameras and LiDAR point clouds. Then we find angles (edges) in images and in point clouds. At the second stage we create a set of respective points. Main part of the second stage is calculation of the transformation matrix, that gives us correlation between camera and LiDAR frames.

In Figure 8 we show how calibration algorithm works for different calibration plate positions. Regardless to plate rotation angle and shifts of the frame, algorithm is capable to find angles precisely.

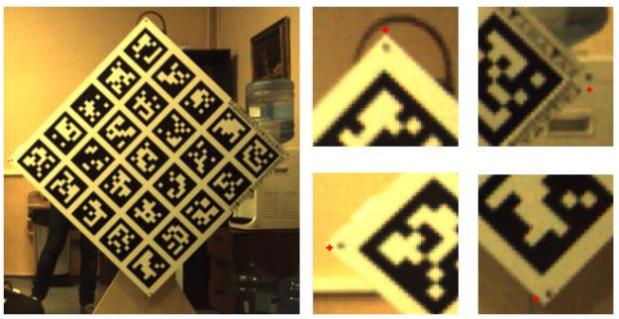


Figure 8. Detected board edges

Detection of the board edges in the LiDAR point cloud is shown in Figure 9.

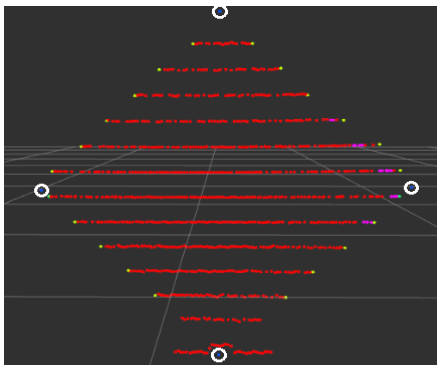


Figure 9. Board edges in LiDAR data

4.4 Calibration results

The result of the orientation procedure directly depends on the quality of the source data, and therefore on the accuracy of determining the board angles in the LiDAR cloud is relatively low. However, making several calibration "attempts" and using RANSAC algorithm (Fischler, Bolles, 1981) to select the best result allows us to get quite low reprojection errors (Table 1).

Test #	Attempt count	Reprojection errors (pixels)			
		Max	Min	Average	Median
1	5	22.27	2.27e-13	9.49	8.70
2	9	30.06	2.27e-13	7.82	6.51
3	13	31.42	0	7.20	5.81
4	17	28.64	0	6.96	5.73

Table 1. Reprojection errors in orientation calculations for different numbers of attempts.

We calibrate every side of the autonomous platform separately. We align the resulting point cloud knowing the relative position of all sensors.

5. SEMANTIC SEGMENTATION

A vision system based on semantic segmentation algorithms is one of the key elements of an off-road autonomous robotic vehicle. Semantic segmentation is used for recognition of the underlying surface type, for calculation of patency map, for detection, recognition and tracking of objects and obstacles. The imposition of semantic segmentation on a three-dimensional model or point cloud gives us the class of each point and adjust the patency map of the robotic vehicle.

Currently, the task of semantic segmentation is being generally solved by using convolutional neural networks, which can take an image of arbitrary size as an input and output an appropriate predict.

5.1 Off-road dataset

Recently, semantic segmentation algorithms have been actively developed due to their application in various fields. Autonomous transport is one of the ways to apply these algorithms.

However, most databases of images, annotated with segmentation masks, were collected in urban street scenes. This implies the presence of buildings, paved roads, sidewalks, pedestrians and many different vehicles. Therefore, for semantic off-road scene understanding we created our original dataset consisting of around 100,000 annotated images in addition to the simulated dataset of 350,000 annotated images, in which we included forests, groups of trees, bushes, embankments, ravines, ditches, stones, fields, various types of dirt roads, buildings, structures and other types of obstacles. It was captured in the countryside at every time of the year, at different times of day, in different weather conditions.

Subsequently, we found the terrain and lighting conditions in which the models predicted wrong labels, so we added more data for them. We defined 14 classes: hard ground, soft ground, building, fence, impassable vegetation, passable vegetation, sky, people, vehicle, water, traffic sign, pole, other obstacles and void.

5.2 CNN Architecture

We propose an architecture (Figure 10) that can handle U-Net (Ronneberger et al., 2015) and DeepLabV3 (Chen et al., 2017) as a decoder inspired by works (Siam et al., 2018), (Gamal et al., 2018) and (Pavel Yakubovskiy, 2020). At the start we also considered Pyramid Attention Network (Li et al., 2018) and Receptive Field Block (Liu et al., 2017), but they either did not work in the real-time, or provided poor quality for the off-road data.

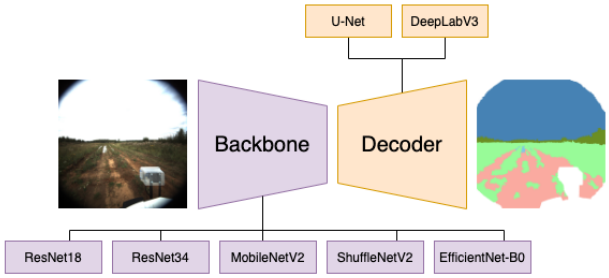


Figure 10. Proposed backbones and decoders

We experimented with Resnet18, Resnet34 (Kaiming He et al., 2015), MobileNetV2 (Mark Sandler et al., 2018), ShuffleNetV2 (Ningning Ma et al., 2018) and EfficientNet-B0 (Mingxing Tan et al., 2019) backbones that demonstrated similar results on the ImageNet dataset. Also, we used various pre-trained weights for train models to compare them and achieve boost of accuracy of semantic segmentation.

Method	Top-1 error (%)
ResNet18	30.2
ResNet34	26.7
MobileNetV2	28.1
ShuffleNetV2	30.6
EfficientNet-B0	23.7

Table 2. Top-1 error on ImageNet

Moreover, used encoders and decoders were reimplemented and adapted for converting to ONNX model and then to NVIDIA TensorRT engine.

Finally, we chose the ResNet34+DeepLabV3 combination, as it gives us real-time processing with a very good performance in terms of quality (see the "Results" paragraph below).

5.3 Results

Datasets were divided into train set (80%) and test set (20%). We evaluate our models on simulated dataset, original off-road dataset and Cityscapes (Cordts et.al, 2016). Intersection-over-union (IoU) metric is used as an assessment of accuracy.

Firstly, we trained models on our simulated dataset that was created on the Unreal Engine 4 and on the Cityscapes train set to get pre-trained weights.

Secondary, we used these pre-trained weights to finetune models on our original off-road dataset to compare results. Using the pre-trained weights on our simulated dataset, model with backbone of ResNet34 and DeepLabV3 decoding increased 2.7% mIoU compared to the pre-trained weights on Cityscapes (see comparison in Table 4 below).

This increase in accuracy allows us to use ResNet34 as encoder instead of more heavyweight backbone such as ResNet50, ResNet101 etc. At the same time, we improve inference time and get a comparable accuracy that is a distinct advantage of this approach.

In Table 3 we present results on Cityscapes validation set. Calculations were made using NVIDIA GeForce RTX 2080.

Method	mIoU	Time(ms)
HRNetV2	81.1%	-
DeepLabv3 (ResNet101+ASPP)	78.5%	491
Our (ResNet34 + DeepLabV3)	**75.6%**	**157**

Table 3. Cityscapes validation set results

In Table 4 we show results on our simulated validation set.

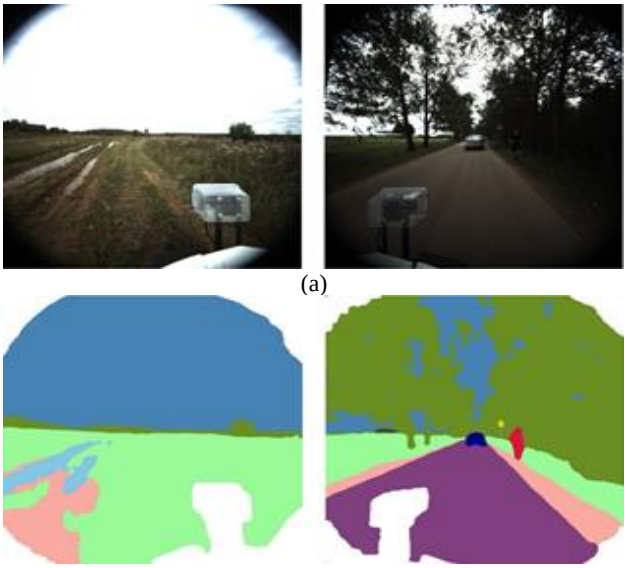
Decoder	Backbone	mIoU (%)
U-Net	ResNet18	68.8
	ResNet34	70.3
	MobileNetV2	67.5
	ShuffleNetV2	67.0
	EfficientNet-B0	65.3
DeeplabV3	ResNet18	95.2
	ResNet34	**95.5**
	MobileNetV2	94.5
	ShuffleNetV2	93.2
	EfficientNet-B0	90.1

Table 4. Our simulated validation set results

In Table 5 we present results on our own validation set of the off-road dataset, consisting only of real images.

Pretrained	Decoder	Backbone	mIoU (%)
Cityscapes	U-Net	ResNet18	62.4
		ResNet34	63.9
		MobileNetV2	60.7
		ShuffleNetV2	57.3
		EfficientNet-B0	54.1
	DeeplabV3	ResNet18	80.8
		ResNet34	82.5
		MobileNetV2	78.1
		ShuffleNetV2	75.3
		EfficientNet-B0	72.8
Our simulated dataset	U-Net	ResNet18	64.0
		ResNet34	66.4
		MobileNetV2	63.5
		ShuffleNetV2	59.1
		EfficientNet-B0	57.1
	DeeplabV3	ResNet18	83.3
		ResNet34	**85.2**
		MobileNetV2	81.0
		ShuffleNetV2	78.7
		EfficientNet-B0	74.9

Table 5. Our off-road validation set results.



(a)

(b)

Figure 11. Original images (a) from our off-road dataset and results of the proposed algorithm (b)

5.4 Optimizations

The inference of neural networks is expected to have minimal latency, maximum throughput, optimal memory consumption usage and power efficiency.

Inference of almost any model can be implemented using deep learning frameworks (Caffe, MxNet, Keras, Tensorflow, PyTorch) and special compiler-optimizers that rebuild the neural network architecture for a hardware device (CPU, GPU, NPU). PyTorch is an extremely useful tool for training neural networks, but it does not provide fastest inference time on GPU. Therefore, we used compiler-optimizer NVIDIA TensorRT, which performs optimization of a neural network for NVIDIA GPU platforms. This tool allows to speeds up the inference time using various optimizations such as vertical and horizontal layer fusion etc.

NVIDIA TensorRT as an input parameter takes a model of a neural network that has been converted from PyTorch to ONNX and serialize engine.

Implementation of this model on NVIDIA GeForce RTX 2080 using NVIDIA TensorRT requires about thrice less time to process in comparison with PyTorch version of this model on a stronger NVIDIA GeForce RTX 2080 Ti (Table 6). Pre-processing and post-processing operations were also performed on GPU.

Method	Time(ms) on PyTorch (NVIDIA 2080 Ti)	Time(ms) on TensorRT (NVIDIA 2080, fp16)
ResNet18 + DeepLabV3	56	22
ResNet34 + DeepLabV3	**85**	**27**
MobileNetV2 + DeepLabV3	65	54

Table 6. Inference time for a 1,024×1,024 input on PyTorch and NVIDIA TensorRT

6. OBJECT DETECTION

Real-time object detection in autonomous driving systems got serious boost in last years (Zhong-Qiu Zhao et al., 2019). Detecting a pedestrian crossing AV trajectory, as well as potential obstacles are the crucial problems. It is useless to say that detection algorithms must be reliable and robust to all kind of alternations (Zhongmin Liu et al., 2018). These algorithms must be fast enough to work in real time.

A major part of recent practical works in this domain demonstrated adequate results on cityscapes. However, off-road areas make important part of whole environment and remain partially covered by practitioners. Even if savage nature poses additional difficulties for object detection, solution for such circumstances could be used in domains beyond autonomous driving.

A number of recent papers propose object detection algorithm showing remarkable results. Nevertheless, we obtain modest results when apply these algorithms directly to our countryside datasets. Most of the popular datasets used as quality standard are not difficult enough, especially for occlusions and partially visible objects. Considering only minor changes to algorithm and hyperparameters adjustment we conclude that data preparation is the main issue. We argue that thorough data analysis and dataset composition may compensate shortcomings from algorithms elaborated and tested on standard datasets.

Our pipeline is based on RFB Network algorithm. Basis of our DCNN is RFB-block (Songtao Liu et al., 2018), it provides sufficient quality to be used for object detection. On the other hand, it is fast enough to be used for real time detection on board of mobile platform.

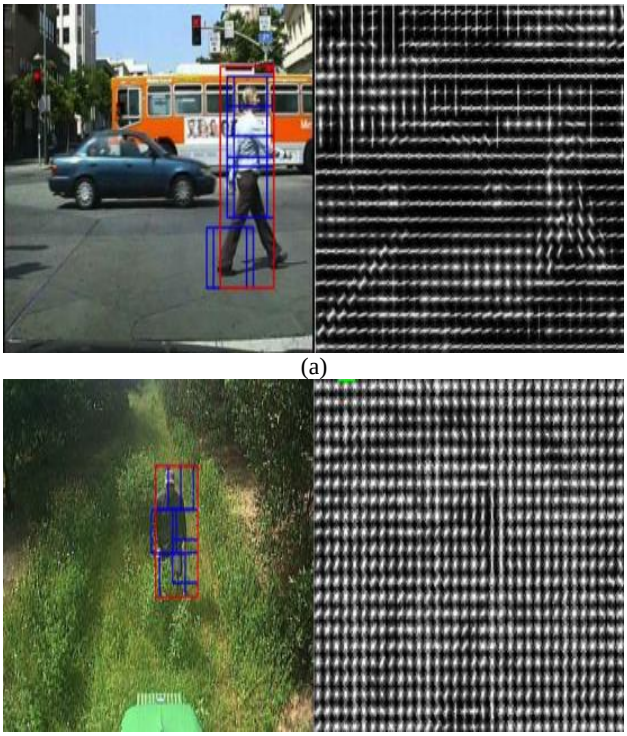
6.1 Special conditions

Nature features within off-road areas lead to particular problems to be solved in object detection domain (Dong-Ki Kim et al., 2017). Even a term “off-road area” is not defined distinctly. The certain thing that it’s not a cityscape or highway, but it’s all the rest. Off-road areas are more prone to landscape seasonal changes.

In northern regions white color prevails in winter time, while it’s not the same in southern. Autumn and Spring have their own particular colors and textures (green grass, yellow leaves, black earth, etc.). Different regions possess their particular textures and gradations. In contrast, cityscapes gammas do not vary a lot. Grey color prevails almost everywhere, excluding some regional particularities.

6.2 Gradients distribution

Within city area a majority of objects possess strict geometrical forms, even trees and bushes are aligned and trimmed. Gradient distribution of background areas on an image taken in city and in off-road landscape differs noticeably (Pezzementi et al., 2017).



(a)

(b)

Figure 12. Gradient distribution in city landscape (a) and in off-road landscape (b)

This is one of the reasons why object detection in cityscapes is not exactly the same problem as off-road object detection, which is shown in Figure 12 (Tabor et al., 2015).

6.3 Savage nature

Natural phenomena like wind does not influence objects in cityscape as much as it does in off-road landscape. Strong wind affects high grass, bushes, trees. It severely changes their shapes and slopes. Thus, it can make changes in other objects on scene or change occlusion sectors.

6.4 Objects poses

While we are looking for a cityscape pedestrian detection problem, we expect people to appear in certain positions on scene, they are walking or standing. Most of datasets contain images with people in mentioned positions. You cannot expect all imaginable positions people can take in off-road scenes: people can be partly occluded by grass or other vegetation. It depends on area, situation, circumstances, it can vary largely.

6.5 Dataset

The dataset was collected in several areas with diverse landscape and at vary seasons and weather conditions. Main problem of datasets is that data is imbalanced in terms of object classes appearance frequency and background homogeneity. To create a balanced, heterogeneous and sufficiently large dataset that generalize all desired object features in all possible conditions is a challenging task that takes time. From a crude data collected from different cameras we obtained about 1,000,000 images. After thorough analysis and refinement, we picked about 50,000 labeled images. The whole set was divided into train and test sets in proportion 90/10.

6.6 Artificial negative mining

As dataset was formed gradually in a period of 6 month. After slowly feeding small collection with mostly clear and distinct objects presented in images, we realized that occasional difficult cases are not being detected. Thus, we created such occasions artificially overlapping human figures by bushes, high grass, encouraging cases of occlusions and intersections. We struggled to present our objects in dataset in all imaginable perspectives. Finally experiments with lighting either natural or artificial and camera adjustments permitted us to inflate the dataset with unique data.

6.7 Implementation

We implemented whole algorithm on board of out autonomous platform. We were limited in CPU and GPU computing resources, as object detection algorithm is just a part of the whole system that should work in real-time, providing sufficient quality. Since we use computing unit with the Nvidia RTX2080 graphics card, we chose TensorRT as the inference framework. It is a C++ library that facilitates high-performance computations on Nvidia GPUs compared to other inference frameworks. TensorRT optimizes the network by combining layers and optimizing kernel selection for improved latency, throughput, power efficiency, and memory consumption. We transform our model, trained using PyTorch, into ONNX format that is supported by TensorRT. We created our own high-level library to perform all image processing operations (resizing, transposition, channel swap etc.) on GPU. Finally, the whole algorithm shows 38 FPS.

6.8 Training and results

Our network is implemented with PyTorch. The batch size is 16 per GPU, optimizer is Adam with default parameters. We trained our network for 100 epochs with learning rate 0.001, downgrading it 10 times for each 100 epochs. You can see pedestrian detection results in the Figures 13-15.

Figure 13. Detection results

Figure 14. Detection results

Figure 15. Detection results

7. CONCLUSIONS

In this paper we described several methods of a complex approach to autonomous driving problem in off-road conditions, in the countryside is still a challenging problem. Our solutions are based on robust algorithms of sensors calibration, semantic segmentation, three-dimensional scene reconstruction, object detection.

We created a large database containing field data (more than 1,000,000 images and LiDAR data), as well as virtual data (more than 3,500,000 images and simulated LiDAR data).

ACKNOWLEDGEMENTS

The reported study was funded by RFBR project № 19-07-01248 A.

REFERENCES

Chai, Z., Sun, Y., Xiong, Z. A, 2018. Novel Method for LiDAR Camera Calibration by Plane Fitting // In Proceedings of the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand, pp. 286–291.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, 2016. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.

L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. arXiv:1706.05587v3 [cs.CV]

M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding // In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Dhall A, Chelani K, Radhakrishnan V, et al., 2017. LiDAR-Camera Calibration using 3D-3D Point correspondences. arXiv:1705.09785 [cs.RO]

M. A. Fischler and R. C. Bolles, 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. Of the ACM*, vol. 24, 381-395, 1981.

M. Gamal, M. Siam, M. Abdel-Razek, 2018. ShuffleSeg: Real-time Semantic Segmentation Network.

Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ, 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47(6). pp. 2280-2292.

Guindel, C., Beltrán, J., Martín, D. and García, F., 2017. Automatic Extrinsic Calibration for Lidar-Stereo Vehicle Sensor Setups // IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 674–679.

K. He, X. Zhang, S. Ren, J. Sun, 2015. Deep Residual Learning for Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778.

D-K Kim., D Maturana, M Uenoyama, S Scherer, 2017. Season-invariant semantic segmentation with a deep multimodal network. *Field and Service Robotics*, pp. 335-350.

H. Li, P. Xiong, J. An, L. Wang, 2018. Pyramid Attention Network for Semantic Segmentation.

Liu Z., Chen Z., Li Z., Hu W., 2018: Mathematical Problems in Engineering Volume 2018, Article ID 3518959, 10 pages. An Efficient Pedestrian Detection Method Based on YOLOv2.

Liu S., Huang D., Wang Y, 2018. Receptive Field Block Net for Accurate and Fast Object Detection. In: Computer Vision – ECCV 2018. Lecture Notes in Computer Science, vol 11215. Springer, Cham.

S. Liu, Di Huang, Y. Wang, 2017. Receptive Field Block Net for Accurate and Fast Object Detection. arXiv:1711.07767 [cs.CV]

N. Ma, X. Zhang, H.-T. Zheng, J. Sun, 2018. ShuffleNet: ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design // In ECCV 2018. Lecture Notes in Computer Science, vol 11218. Springer, Cham.

Y. Park, S. Yun, C.S. Won, K. Cho, K. Um, S. Sim, 2014. Calibration between color camera and 3D LIDAR instruments with a polygonal planar board // Journal of Sensors. Vol. 14, Issue 3, pp. 5333-5353.

Pereira M, Silva D, Santos V, et al., 2016. Self-calibration of multiple LIDARs and cameras on autonomous vehicles // Robotics & Autonomous Systems, 83(C), pp. 326-337.

Pezzementi Z., Tabor T., Hu P., Chang J.K., 2017: Comparing Apples and Oranges: Off-Road Pedestrian Detection on the NREC Agricultural Person-Detection Dataset.arXiv:1707.07169, 2017 [cs.CV]

Z. Pusztai, L. Hajder, 2017. Accurate calibration of LiDAR-camera systems using ordinary boxes // 2017 IEEE International Conference on Computer Vision Workshops (ICCVW) – pp. 394-402.

F. J. Romero-Ramirez, R. Muñoz-Salinas, R. Medina-Carnicer, 2018. Speeded up detection of squared fiducial markers // Image and Vision Computing, vol 76, pp. 38-47.

O. Ronneberger, P. Fischer, and T. Brox, 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation.

M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, 2018. MobilenetV2: Inverted residuals and linear bottlenecks // 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510-4520.

M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, M. Jagersand, 2018. RTSeg: Real-time Semantic Segmentation Comparative Study.

T. Tabor, Z. Pezzementi, C. Vallespi and C. Wellington, 2015. People in the weeds: Pedestrian detection goes off-road, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), West Lafayette, IN, pp. 1-7.

M. Tan, Q. V. Le, 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946v3 [cs.LG]

Xu Z, Li X, 2014. A method of extrinsic calibration between a four-layer laser range finder and a camera // IEEE Control Conference, pp. 7450-7455.

P. Yakubovskiy, 2020. Segmentation Models PyTorch. https://github.com/qubvel/segmentation_models.pytorch

Zhao Z-Q., Zheng P., Xu S-T., Wu X., 2019: Object Detection with Deep Learning: A Review, arXiv:1807.05511 [cs.CV]