2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Online Camera LiDAR Fusion and Object Detection on Hybrid Data for Autonomous Driving

Koyel Banerjee[1], Dominik Notz[1], Johannes Windelen[1], Sumanth Gavarraju[2] and Mingkang He[2]

Abstract— **Environment perception for autonomous driving traditionally uses sensor fusion to combine the object detections from various sensors mounted on the car into a single representation of the environment. Non-calibrated sensors result in artifacts and aberration in the environment model, which makes tasks like free-space detection more challenging. In this study, we improve the LiDAR and camera fusion approach of Levinson and Thrun. We rely on intensity discontinuities and erosion and dilation of the edge image for increased robustness against shadows and visual patterns, which is a recurring problem in point cloud related work. Furthermore, we use a gradient-free optimizer instead of an exhaustive grid search to find the extrinsic calibration. Hence, our fusion pipeline is lightweight and able to run in real-time on a computer in the car. For the detection task, we modify the Faster R-CNN architecture to accommodate hybrid LiDAR-camera data for improved object detection and classification. We test our algorithms on the KITTI data set and locally collected urban scenarios. We also give an outlook on how radar can be added to the fusion pipeline via velocity matching.**

I. INTRODUCTION

Autonomous cars rely on a variety of sensors to perceive their environment. In order to build a consistent model of their surrounding world, which is needed to act safely in it, the data of the different sensors needs to be fused [1]. Each type of sensor comes with its own strengths and weaknesses [2]. RGB cameras perceive color and texture information from the world and are good for object classification tasks. However, their detection range is limited and they perform poorly in limited lighting or adverse weather conditions. LiDARs provide precise distance information, have ranges that can exceed 100m and are able to detect small objects. They also work well at night but do not provide color information and their performance decreases in heavy rain [3] [4]. Radars provide precise distance and velocity information and work well in inclement weather conditions but have a rather low resolution [5].

These days high-level fusion (HLF) is a very popular sensor fusion approach [1]. HLF detects objects with each sensor separately and subsequently combines these detections. Hence, object detections are made locally with limited available information because HLF discards classifications with low confidence values, if for example, there are multiple overlapping objects and artifacts. In contrast, low-level fusion (LLF) combines the data from different sensor types at the raw data level, thereby preserving all information and potentially increasing the object detection accuracy. LLF is intrinsically complex and comes with several challenges. A very accurate extrinsic calibration of the sensors is needed to correctly fuse their perceptions of the environment. In addition, the sensor recordings need to be time-synchronized and compensated for ego-motion. The multi-modal input data can then be used for object detection with deep neural networks. Both the fusion and detection algorithms must to be capable to run in real-time in an on-road driving scenario. Due to its promising advantages LLF has attracted some attention, recently. For instance, Chen et al. [6] have developed a fusion approach for LiDAR and camera data, which outperformed existing methods in 3D localization and 3D detection on the KITTI data set [7]. Levinson and Thrun [8] developed an algorithm for the extrinsic calibration of LiDAR and camera but do not use the fused data for object detection. A promising method that depends on 3D target based calibration was developed by Geiger et al. [9] using multiple checkerboards on the same scene and using a trinocular camera with range sensors or Microsoft Kinect.

For our LLF approach, we improve the extrinsic LiDAR camera calibration method of Levinson and Thrun [8]. We apply erosion and dilation to the inverse distance transformed image edges. We found that these morphologies smooth the objective function and improve the robustness to shadows and other visual patterns in the scene. The qualitative comparison between our approach and the KITTI supplied extrinsic calibrations shows that our method yields a visually better fit. Moreover, we replace the exhaustive grid search over the transformation parameters with a gradient-free optimization algorithm. We project the LiDAR point cloud onto the RGB image, upsample it and subsequently extract features from it. We adapt the Faster R-CNN architecture [10] to operate on the fused input data. Our results show that using the fused LiDAR and camera data improves the object detection results on the KITTI data set [7]. Our sensor fusion algorithm is able to run in real-time with 10Hz. Furthermore, we give an outlook on a novel approach for the extrinsic calibration between LiDAR and radar.

The rest of the work is arranged as follows. In Section II, we provide an overview of related work. In Section III, we derive our algorithms. First, we discuss the fusion of LiDAR and camera. Subsequently, we present our modifications of the Faster R-CNN architecture to leverage the fused data. In Section IV, we discuss our results and finally give a summary and an outlook on future work.

[1]Koyel Banerjee (koyel.banerjee@bmw.de), Dominik Notz (dominik.notz@bmw.de) and Johannes Windelen (johannes.windelen@bmw.de) are with the BMW Group Technology Office.

[2]Sumanth Gavarraju (sumanth.gavarraju@bmw.de) and Mingkang He (mingkang.he@bmw.de) were interns with the BMW Group Technology Office.

II. RELATED WORK

The computation of the extrinsic calibration between camera and LiDAR usually depends on pre-defined markers, such as checker boards or AR tags, and the calibration process itself is mostly offline. The approach by Dhall et al. [11] uses special encoded ArUco markers [12] to obtain 3D-3D point correspondences between LiDAR and camera. The calibration method of Velas et al. [13] also relies on specific 3D markers. In [14] Schneider et al. propose a deep learning based end-to-end architecture for feature extraction, feature matching and global regression. Their approach is able to run in real-time and can adjust for online mis-calibration errors. However, a large amount of data is required to train the system from scratch. Also, a separate data collection for each vehicle would be necessary as sensor alignments and intrinsics might vary. The work by Castorena et al. [15] is based on edge alignments between optical camera and LiDAR data using reflectivity values. Using the KITTI extrinsic parameters as ground truth, they achieve near KITTI precision for a target-less approach. The target-less camera LiDAR calibration approach of Levinson and Thrun [8] fits edges in image and point cloud data but uses an exhaustive grid search. Our approach of extrinsic calibration builds upon it but works online and does not need an exhaustive computational search over parameters.

III. METHODOLOGY

This section gives an overview of current research pertaining to the fusion of camera and LiDAR sensor data, consisting of intrinsic sensor calibration, extrinsic calibration between the sensors and data fusion to create a hybrid data type. Sensor fusion can be divided into three categories: LLF, mid-level fusion (MLF) and HLF. HLF has been the most popular fusion technique with car OEMs mainly because it uses the vendors' supplied object lists from the sensors and integrates them into an environment model [1]. However, since the sensors are not calibrated against each other this method causes aberrations and duplicate objects. One way to prevent the occurrence of these issues is the fusion of raw sensor data (LLF). MLF is an abstraction sitting on top of LLF, where extracted features from multiple sensor data are fused. Figure 1 shows the sensor setup on our car. An overview of the entire fusion pipeline is given in Figure 2. The following sections explain each part in more detail.

A. *Fusion of LiDAR and Camera*

In this section, we describe our approach for the extrinsic calibration of LiDAR and camera, the projection of the LiDAR point cloud (PC) onto the image, its upsampling and feature extraction from the depth map. In order to ensure that we capture a PC and an image at the same time we trigger the recording of both sensors via software. We use a Velodyne HDL-64E S3 as our LiDAR sensor, which rotates with a frequency of 10Hz. In order to reduce the distortion of the captured PC we compensate for the ego-motion of the vehicle during the sensor recordings.

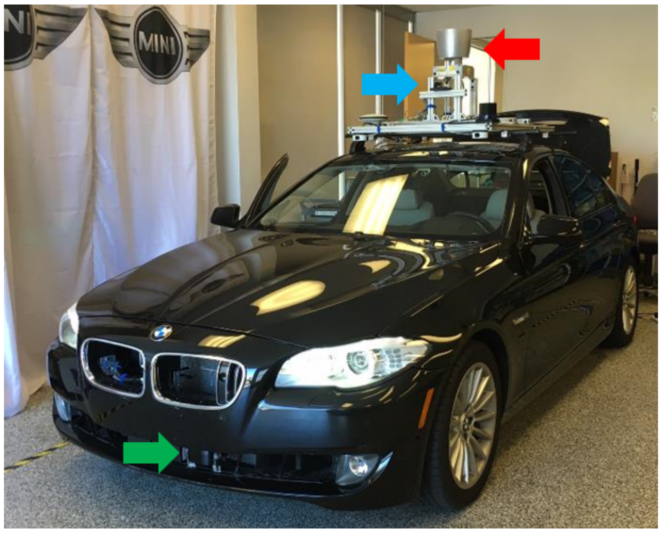

Fig. 1. The sensor setup of our BMW test vehicle consists of a LiDAR (red arrow), a camera (blue arrow) and a radar (green arrow).

The extrinsic sensor calibration relies on accurate intrinsic calibrations of the sensors. The 3×4 camera intrinsic matrix ${}^{I}\mathbf{T}_{C}$ defines the projection in homogeneous coordinates from the camera to the image coordinate system. We follow the approach of Datta et al. [16] and use a ring pattern calibration board (see Figure 3) with iterative refinement of the parameters. Compared to the standard camera calibration method of Zhang with a checkerboard [17] we observed a 70% reduction of the average reprojection error. The intrinsic LiDAR calibration, which defines the transformations from each emitter to the sensor base coordinate system, is provided by the manufacturer.

The extrinsic calibration between LiDAR and camera corresponds to finding the 4×4 transformation matrix ${}^{C}\mathbf{T}_{L}$ between their coordinate systems. ${}^{C}\mathbf{T}_{L}$ consists of a rotation and a translation and hence has six degrees of freedom (DoFs) (see Figure 4). For finding the extrinsic calibration between LiDAR and camera, we generally follow the approach of Levinson and Thrun [8] but make several adaptions to it, as noted in the following paragraphs. The underlying concept is to find the six parameters defining ${}^{C}\mathbf{T}_{L}$ such that edges in the camera image match discontinuities in the point cloud measurements. For that reason, we define a similarity function S. We perform an element-wise multiplication of the depth discontinuities of the projected point cloud image X with an edge image E and return the sum of this product over all pixels i. In order to smooth the objective function and cover a larger variety of scenes we use N pairs of point clouds and images and sum their matching scores. The objective is to find the transformation ${}^{C}\mathbf{T}_{L}$ which maximizes

$$S(\mathbf{T}) = \sum_{f=1}^{N} \sum_{i} X_i^f(\mathbf{T}) \cdot E_i^f . \tag{1}$$

The edge image E is obtained as follows. We convert the RGB image to grayscale and compute its edges with the Sobel operator. In order to also reward nearly matching PC

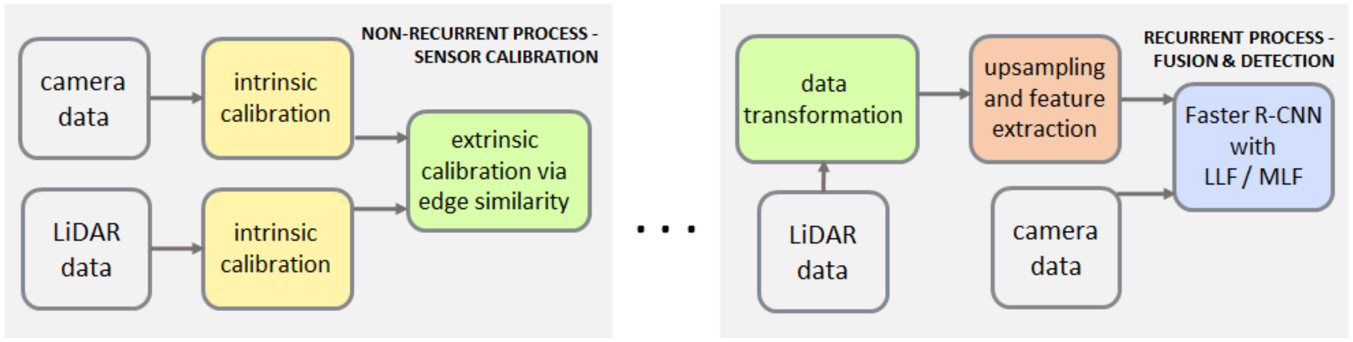
NON-RECURRENT PROCESS - SENSOR CALIBRATION
camera data
intrinsic calibration
extrinsic calibration via edge similarity
LiDAR data
intrinsic calibration
RECURRENT PROCESS - FUSION & DETECTION
data transformation
upsampling and feature extraction
Faster R-CNN with LLF / MLF
LiDAR data
camera data

Fig. 2. Overview of our sensor fusion and object detection pipeline. The left diagram visualizes the extrinsic calibration between the sensors and is only executed once or when a new calibration is desired. The right diagram shows the pipeline of the periodically running sensor fusion and object detection / localization.

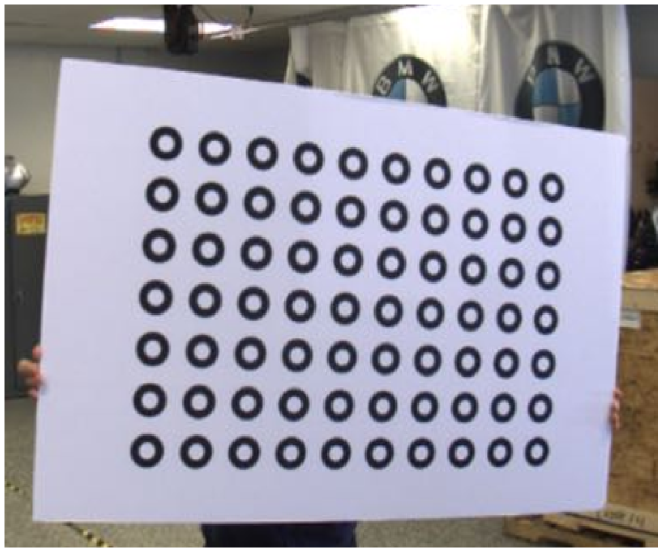

Fig. 3. Ring pattern board used for camera calibration. Following the intrinsic camera calibration approach in [16] reduced the average re-projection error by $\sim 70\%$ compared to the standard calibration method from [17].

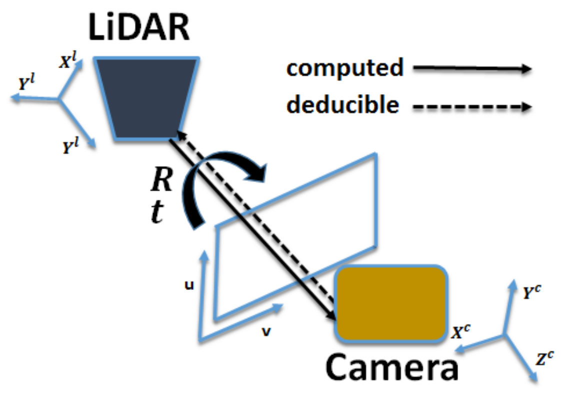

Fig. 4. Problem statement of the extrinsic calibration of LiDAR and camera. Computing the extrinsic calibration between two sensors refers to estimating the rotation R and translation t between their coordinate systems. We explicitly compute the extrinsics from LiDAR to camera and are able to deduce the other direction.

and image edges we blur the image edges. Levinson and Thrun [8] use the inverse distance transformation (IDT) for that purpose. Another possibility is to apply a Gaussian filter. We found that combining IDT with erosion and dilation (ED) gave the best results. IDT+ED was found to increase the robustness to shadows in the scene, which are very common in driving scenarios. Figure 5 exemplary visualizes why IDT+ED is advantageous. In this image, ED has not been applied. The small shadows from the trees in the center and below the car on the right hand side of the image result in image edges, for which no corresponding edges in the PC exist. Nevertheless, the optimization method will try to match these edges resulting in a worse calibration estimate. In this case, the absence of ED yields a calibration error which can be seen at the tires of the vehicle on the right hand side. ED is able to reduce the influence of such small texture differences in the images. As in [8], we extract the

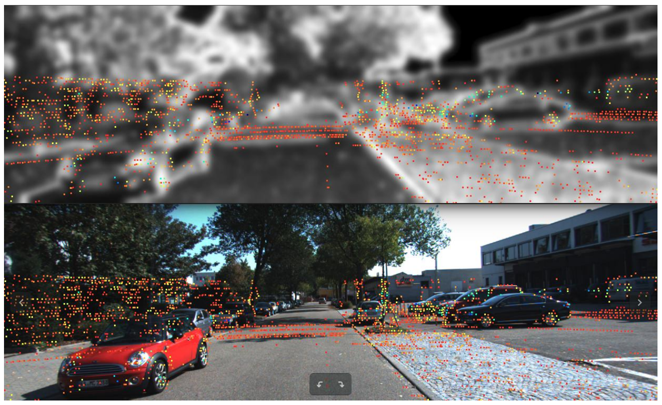

Fig. 5. Non-optimal calibration estimation due to the left out application of erosion and dilation to the extracted edge image. The top (bottom) image shows an extracted edge image to which ED has not been applied (RGB image) overlaid by the projected PC based on the calibration estimate. Small shadows in the RGB image from the trees in the center and below the car on the right result in edges, for which no corresponding edges in the PC data exist. Hence, the optimization method will yield a non-optimal calibration estimate, which can be seen at the tires of the car on the right hand side. ED is able to lessen the influence of such small textures.

range discontinuities in the PC P, resulting in a point cloud of discontinuities P^* with

$$P^*_{i,j} = \max(P_{i,j-1} - P_{i,j}, P_{i,j+1} - P_{i,j}, 0). \qquad (2)$$

Other than in [8], $P_{i,j}$ can not only refer to the depth but also to the intensity of the j^{th} measurement of beam i. We found that the calibration results obtained using the intensity values were better, since planar surfaces with edges and different materials do not exhibit range but intensity discontinuities. The newly obtained PC is then transformed with an estimate of ${}^{C}\mathbf{T}_L$ to the camera coordinate system and subsequently projected with the camera intrinsic matrix ${}^{I}\mathbf{T}_C$ onto the image plane, yielding the projected point cloud image X.

The similarity function S is non-convex and cannot be solved analytically. Levinson and Thrun [8] use a computationally expensive exhaustive grid search over the six parameters for an initial calibration, which they then gradually adjust online. Due to our modifications of their method as described above we are able to use a gradient-free optimization method starting from an initial guess of the parameters to find the optimal ${}^{C}\mathbf{T}_L$. We use BOBYQA [18], an iterative algorithm for finding the minimum of a blackbox function subject to bounds of the optimization variable. BOBYQA relies on a quadratic approximation with a trust region. We were able to find the optimal parameters starting from initial guesses off by several centimeters / degrees. Hence, our method is able to account for manufacturing tolerances.

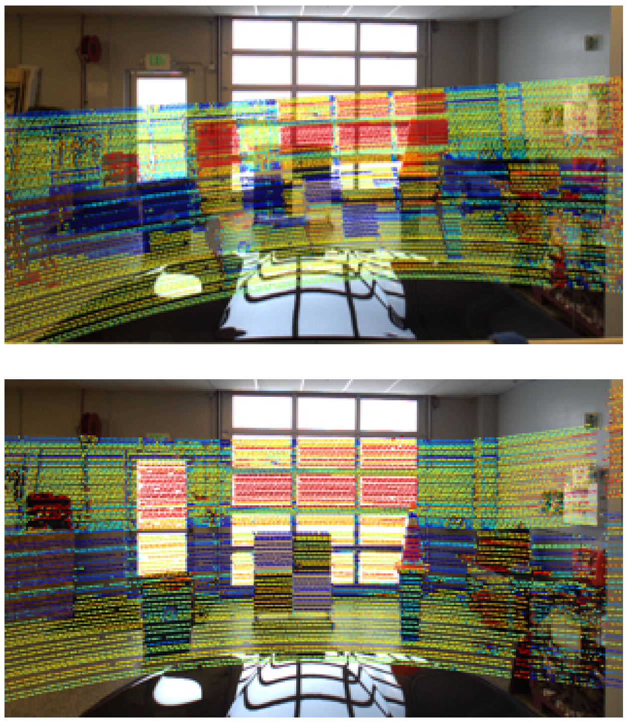

Fig. 6. Comparison of an image overlaid with the intensity values of a PC with matching and mismatching extrinsic LiDAR camera calibrations. The top image shows a mismatch in the LiDAR camera extrinsics with misaligned image and PC edges. In the bottom image the extrinsics are matched well and the edges align well.

It's possible to track the correctness of the extrinsic calibration estimate ${}^{C}\mathbf{T}_L$ as shown in [8] by making use of the local convexity around the correct extrinsic calibration. The idea is to analyze whether the current calibration C results in a local maximum S_c of S. Ideally, if the given calibration C is correct then any small deviation from C should lower the similarity score. We perform a grid search with unit radius centered around a given calibration C across all 6 dimensions yielding $3^6 = 729$ different similarity scores S. One of these will be S_c itself in the center of the grid. F_C is the fraction of the other 728 generated values of S lower than S_C. If the extrinsic calibration C is correct most of these 728 values should be less than S_C, resulting in a F_C near 1. Figure 7 shows the percentage of calibrations

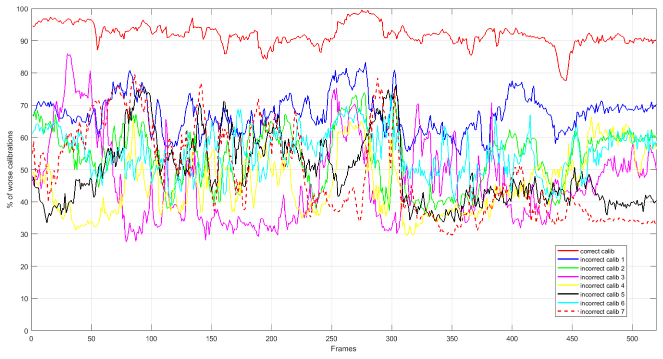

Fig. 7. Percentages of calibrations around a given calibration estimate with lower similarity scores than the similarity score of the actual estimate over different frames. The red curve corresponds to F_C for the correct calibration over multiple frames whereas all other curves correspond to incorrect calibrations.

with similarity scores lower than S_C for a given calibration, resulting in a plot of F_C over a series of frames for different calibration estimates. We can see that the correct extrinsic calibration corresponds to an F_C which is greater than the ones of incorrect calibrations.

The PC is transformed with ${}^{C}\mathbf{T}_L$ and subsequently projected with ${}^{I}\mathbf{T}_C$ onto the RGB image yielding an RGBD image, where depth is sparse. Figure 6 shows visualizations of such RGBD images with matching and non-matching extrinsic calibrations. For the training and evaluation of a neural network on the fused data later on we will need labeled data. As labeling is expensive, we want to make use of weights from a pre-trained network. For image data this can be easily done as lots of labeled data is freely available. Unfortunately, for projected LiDAR data this is not the case. Hence, we aim at encoding the sparse depth map in a way that it resembles feature from an RGB image. This encoding scheme allows us to also use pre-trained network weights from image data for the depth channels in the network. We achieve such an encoding by first upsampling the sparse depth map using a bilateral filter proposed by Premebida et al. [19] yielding a dense depth map. From the dense depth map we can then extract image-like features. One set of such features results from a jet coloring proposed by Eitel et al. [20]. A three-channel encoding is simply obtained by applying a jet color map to normalized depth values. Furthermore, we also extract HHA (horizontal disparity, height above ground, angle to gravity) features [21]. Both sets of three-channel features exhibit similar structures to RGB data. The dimensions of the encoded depth data match the ones of the camera data. RGB combined with JET / HHA yields a total of six data channels, representing our fused data.

B. Object Detection on Fused Data

The basic idea of LLF and MLF is to leverage more distinct and discriminatory feature sets from the multi-sensory fused data, which can improve the detection and classification accuracy. For LLF, we use the standard Faster R-CNN pipeline with VGG16 [10] and modify its input layers to accomodate 6-channel input data. For MLF, we duplicate the first four convolutional layers of the Faster R-CNN network and use one separate branch for each camera and LiDAR data processing. We concatenate the feature vectors after the fourth convolution layer from each branch and input it to the upper part of the standard Faster R-CNN architecure. We use transfer learning [22] and initialize the weights from RGB-only networks onto each sub network for the MLF-based detections. For LLF, we keep the number of convolutional filters identical to the original Faster R-CNN network. For MLF, the number of parameters in the first four layers is doubled due to the two branches.

The detection and classification is performed for the classes ’Car’ and ’Pedestrian’. In our experiments, we use three Faster R-CNN network architectures for RGB-only, RGB-Depth LLF and RGB-Depth MLF. RGB-only is the standard Faster R-CNNN architecture and uses only camera data as an input. We train RGB-Depth LLF and RGB-Depth MLF with the fused data with both JET and HHA depth encodings. The input to RGB-Depth LLF consists of the six-channel fused data. The input to RGB-Depth MLF consists of the RGB and depth-encoded data to the respective network branches.

IV. RESULTS

The results section is divided into sensor calibration and object detection and localization results for LLF and MLF. Our methods have been evaluated on both KITTI [7] and a locally collected data set of urban scenarios. Here, we only give the results on the publicly available KITTI data set.

A BMW test vehicle as shown in Figure 1 was used for both recording an internal data set and evaluating the sensor fusion and detection pipeline in practice. The car contains a computer with an Intel Xeon processor used for the fusion and an NVIDIA graphics card for running the neural networks. Sensors used in this study are a Velodyne HDL-64E S3, a Grasshopper2 GS2-GE-50S5C-C camera from Point Grey Research Inc. and an Axis Communications P1214-E network camera. The sensor fusion runs at 10Hz, synchronized with the LiDaR. The object detection inference takes about 250ms per fused data pair. Optimization frameworks, like NVIDIA’s TensorRT [23], could potentially significantly reduce the inference time.

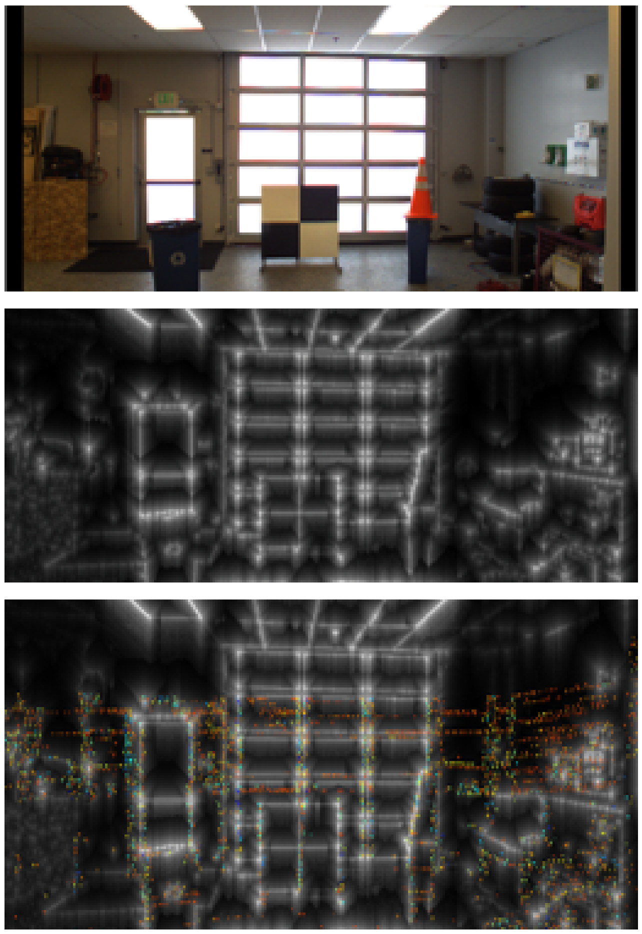

Fig. 8. Visualization of our image edge extraction method. (Top) shows an RGB image, (middle) shows the result of IDT+DE applied to the image edges and (bottom) shows the fitting of the image edges with the PC.

A. Calibration Results

Results of the edge extraction and subsequent edge blurring using IDT+DE are shown in Figure 8. The calibration algorithm successfully converges and is robust to errors in the initial guess. The IDT+DE processing successfully smooths out the gradient in the edge image, resulting in a smoother similarity function, of which the optimizer can find the global maximum. Alternatively, one can use Gaussian smoothing for this purpose which is less accurate but faster. More details were previously explained in Figure 5.

As KITTI is used for the network training, it is essential to have its correct extrinsic calibration. The data set includes an extrinsic calibration, which is calculated from manually selected point correspondences from the PC and the image [24]. We ran our extrinsic calibration method using the KITTI calibration as our initial guess. Table I shows the deviations of our calculated extrinsics from the provided KITTI extrinsics. In general, there is no ground truth for extrinsic calibrations between sensors. Hence, we are unable to provide absolute errors. However, visual inspections (see Figure 9) show that our extrinsics are more accurate than the ones provided by KITTI.

Axis	Rotational Deviations	Translational Deviations
X	0.88°	0.0108m
Y	0.74°	0.0540m
Z	0.93°	0.0529m

TABLE I
ABSOLUTE DIFFERENCES IN THE EXTRINSIC LIDAR CAMERA CALIBRATION PARAMETERS OF KITTI AND OUR OUR METHOD.

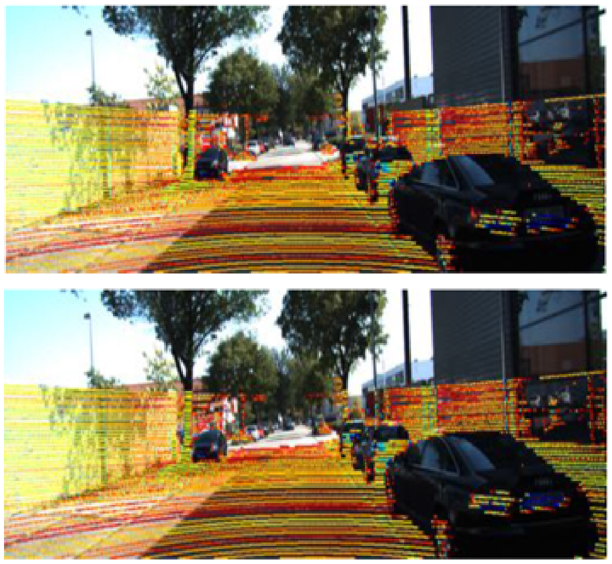

Fig. 9. Comparison of the fused camera and LiDAR data using the extrinsic calibration from KITTI [7] (top) and using the extrinsic calibration computed with our method (bottom).

B. Detection Results

The detection and classification is performed for the classes 'Car' and 'Pedestrian'. We compare the results of RGB-only, RGB-Depth LLF and RGB-Depth MLF architectures with each other and use both JET and HHA depth encodings. We split the KITTI training data set into KITTI-train (70% or about $5,200$ images), KITTI-eval (10% or about 750 images) and KITTI-test (about $1,500$ images). All networks are trained on KITTI-train and evaluated on KITTI-test. It should be noted that we did not perform the evaluation on the official KITTI test set, which is divided into easy, moderate and hard detection problems. Most publications report results on the moderate test set which has some partial occlusions and about 30% truncation. The test portion that we use for our tests is comparable. The car and pedestrian detections on the moderate set for Faster R-CNN are about 79% and 66%. The results of our RGB-only network are similar as shown in Table II. Examples of depth representations with detections are shown in the Figure 10.

The average precision (AP) metric from the Pascal VOC challenge [25] is used to quantify the network's object detection performance. AP combines true positives, false positives and false negatives for a given category into a single metric. We use intersection over union (IoU) as a measure

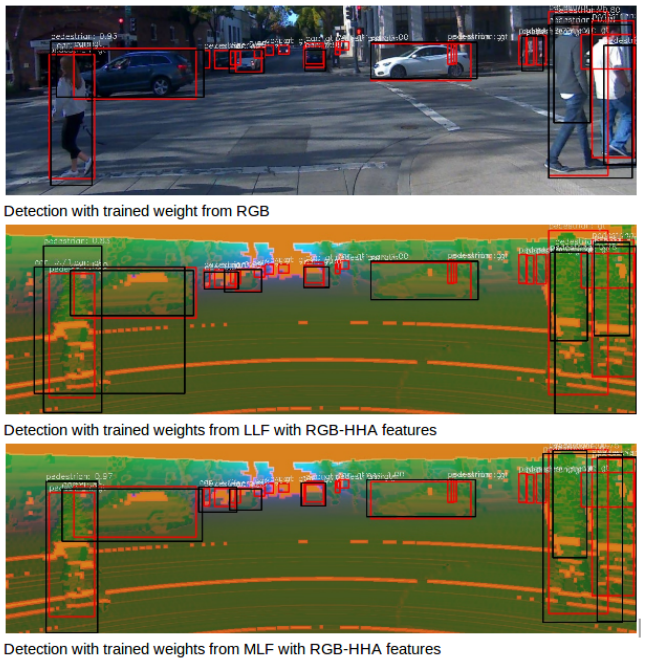
Detection with trained weight from RGB
Detection with trained weights from LLF with RGB-HHA features
Detection with trained weights from MLF with RGB-HHA features

Fig. 10. Visualization of the detection results of cars and pedestrians using HHA depth encoding, Red bounding boxes represent the ground truth and black bounding box represent the detections.

of the overlap between projected and ground truth object locations. In [25] an IoU threshold of 0.5 is used. We set the IoU threshold for a true positive detection to 0.7 to obtain cleaner and more confident results. Although the networks are trained on all classes available in the KITTI data set, only 'Car' and 'Pedestrian' are used in the testing phase as our private data set only contains labels for these classes. The network's performance is evaluated during training to detect over-fitting. It is performed every 100 training iterations and consists of a forward pass of the KITTI-eval data set. Training is halted when the mean average precision (mAP) over all classes on the evaluation reaches its maximum. The network weights of this particular training iteration are then used for the testing phase. The losses of all network architecture trainings converged after around $20,000$ steps. The convergence of the training losses at similar numbers of iterations for all architectures suggests that using the pre-trained RGB weights for the depth channels of the input data is valid - the depth channels contain a similar structure to the RGB channels and thus little fine-tuning of the weights is necessary. The APs from running inference on KITTI-test are shown in Table II. Note that these results cannot be readily compared to entries in the KITTI Vision Benchmark Suite [24] since our evaluation was not performed on the KITTI test data set for which labels are not publicly available. The performance of all networks on the test set is very similar for the 'Car' class. However, the fusion networks perform better on the 'Pedestrian' class, which suggests that the additional depth information is beneficial for pedestrian detection and localization.

Network	Car	Pedestrian
RGB_Faster-RCNN	0.873	0.658
LLF_JET_Faster-RCNN	0.874	0.678
LLF_HHA_Faster-RCNN	0.875	0.695
MLF_JET_Faster-RCNN	0.878	0.703
MLF_HHA_Faster-RCNN	**0.879**	**0.714**

TABLE II
NETWORK EVALUATION RESULTS (AP) ON THE KITTI TEST SET. MLF_HHA_FASTER-RCNN OUTPERFORMS THE OTHER METHODS.

V. CONCLUSION AND FUTURE WORK

In this study we improve the existing calibration method of Levinson and Thrun [8] for better detection and localization accuracy of objects. We improve in a number of ways by using intensity discontinuities from the LiDAR, IDT+DE and a gradient-free optimizer for estimating the rotation and translation parameters. The popular method of fusing high-level object lists from LiDAR and camera lacks proper extrinsic calibration and hence creates aberrations and ringing in the fused data. Our extrinsic calibration method yields results which are more accurate than the calibrations from the KITTI dataset. Our fusion runs in real-time and is lightweight. We upsample the projected point cloud and use different depth encodings (HHA/JET). We show detection and classification results on the low- and mid-level fused RGB and depth data for the camera LiDAR portion.

Our work can be extended by integrating radar into the fusion pipeline. We are currently following an approach for the extrinsic calibration between LiDAR and radar which is based on velocity tracking. In both LiDAR and radar data, measurements belonging to a single object are clustered and then associated with each other based on estimated / measured velocities. The correspondences then yield an estimate of the transformation matrix. Several steps in this approach, the tracking of LiDAR data, the clustering and the contour preserving need to be automated and improved upon.

Furthermore, we expect improved results from the use of upcoming Flash LiDARs. Flash LiDARs will help to eliminate many of the ego-motion based anomalies in the compensated PC and can also aid in better time synchronization between sensors.

REFERENCES

[1] M. Aeberhard and N. Kaempchen, "High-level sensor data fusion architecture for vehicle surround environment perception," in *Proc. 8th Int. Workshop Intell. Transp*, 2011.

[2] J.-r. Xue, D. Wang, S.-y. Du, D.-x. Cui, Y. Huang, and N.-n. Zheng, "A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 122–138, 2017.

[3] J. Lu, H. Sibai, E. Fabry, and D. A. Forsyth, "NO need to worry about adversarial examples in object detection in autonomous vehicles," *CoRR*, vol. abs/1707.03501, 2017. [Online]. Available: http://arxiv.org/abs/1707.03501

[4] "Velodyne64," http://velodynelidar.com/lidar/products/manual/HDL-64E\%20S3\%20manual.pdf, [Online; accessed Nov 15, 2017].

[5] "Continental radar user manual," https://www.continental-automotive.com/getattachment/bffa64c8-8207-4883-9b5c-85316165824a/Radar-PLC-Manual-EN.pdf.aspx, [Online; accessed Nov 15, 2017].

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE CVPR*, 2017.

[7] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.

[8] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: Science and Systems*, 2013, pp. 24–28.

[9] A. Geiger, F. Moosmann, . Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3936–3943.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[11] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "Lidar-camera calibration using 3d-3d point correspondences," *CoRR*, vol. abs/1705.09785, 2017. [Online]. Available: http://arxiv.org/abs/1705.09785

[12] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[13] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of rgb camera with velodyne lidar," in *Comm. Papers Proc. International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2014, pp. 135–144.

[14] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," *CoRR*, vol. abs/1707.03167, 2017. [Online]. Available: http://arxiv.org/abs/1707.03167

[15] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of lidar and optical cameras via edge alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 2862–2866.

[16] A. Datta, J.-S. Kim, and T. Kanade, "Accurate camera calibration using iterative refinement of control points," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1201–1208.

[17] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[18] M. J. Powell, "The bobyqa algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 2009.

[19] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and dense lidar data," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4112–4117.

[20] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 681–687.

[21] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[22] Y. Wei, Y. Zhang, and Q. Yang, "Learning to transfer," *CoRR*, vol. abs/1708.05629, 2017. [Online]. Available: http://arxiv.org/abs/1708.05629

[23] "Nvidia tensorrt," https://developer.nvidia.com/tensorrt, [Online; accessed Nov 15, 2017].

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[25] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4