INFORMATION ACQUISITION ON PEDESTRIAN MOVEMENTS IN URBAN TRAFFIC WITH A MOBILE MULTI-SENSOR SYSTEM

Björn Borgmann[a,b]*, Marcus Hebel[a], Michael Arens[a], Uwe Stilla[b]

[a] Fraunhofer IOSB, Ettlingen, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation.
Fraunhofer Center for Machine Learning. Gutleuthausstr. 1, 76275 Ettlingen, Germany
(bjoern.borgmann, marcus.hebel, michael.arens)@iosb.fraunhofer.de
[b] Photogrammetry and Remote Sensing, Technische Universitaet Muenchen. Arcisstr. 21, 80333 Munich, Germany
stilla@tum.de

KEY WORDS: Mobile laser scanning, LiDAR, pedestrian detection, multi-sensor, urban traffic

ABSTRACT:

This paper presents an approach which combines LiDAR sensors and cameras of a mobile multi-sensor system to obtain information about pedestrians in the vicinity of the sensor platform. Such information can be used, for example, in the context of driver assistance systems. In the first step, our approach starts by using LiDAR sensor data to detect and track pedestrians, benefiting from LiDAR's capability to directly provide accurate 3D data. After LiDAR-based detection, the approach leverages the typically higher data density provided by 2D cameras to determine the body pose of the detected pedestrians. The approach combines several state-of-the-art machine learning techniques: it uses a neural network and a subsequent voting process to detect pedestrians in LiDAR sensor data. Based on the known geometric constellation of the different sensors and the knowledge of the intrinsic parameters of the cameras, image sections are generated with the respective regions of interest showing only the detected pedestrians. These image sections are then processed with a method for image-based human pose estimation to determine keypoints for different body parts. These keypoints are finally projected from 2D image coordinates to 3D world coordinates using the assignment of the original LiDAR points to a particular pedestrian.

1. INTRODUCTION

Pedestrians belong to the group of particularly vulnerable road users, especially in an urban environment where pedestrians and vehicles have to share the traffic space. Therefore, it is helpful for the development of driver assistance systems, for autonomous driving, but also for traffic or urban planning to have automatic capabilities for immediate or long-term information acquisition about pedestrians in the vicinity of vehicles and in urban areas. Among the sensor equipment discussed for future vehicles, a number of established sensor technologies can be found. These include cameras for visible or infrared light, LiDAR sensors, or RADAR. The aforementioned sensor technologies have quite different characteristics with their respective advantages and disadvantages. Visible light cameras, for example, rely on external light sources, which limits their ability to be used at night. LiDAR sensors constitute their own light source and are able to directly acquire three-dimensional geometric information. However, they typically provide a much lower data density than cameras and are not capable of capturing color-based features. This limits their ability to distinguish individual pedestrians based on their appearance and hampers the determination of detailed features, such as the pose of different body parts of a pedestrian. Multi-sensor systems, i.e., vehicles equipped with different types of sensors, ideally allow the advantages of the different sensor types to be combined to provide as much information as possible about pedestrian activity in the vehicle's vicinity.

This paper presents an approach which uses data acquired by multiple LiDAR sensors and cameras on a mobile multi-sensor system to gather information about pedestrians. The LiDAR sensors are used for the initial detection and the tracking of pedestrians, while the cameras are employed to provide data about their body pose.

* Corresponding author

2. RELATED WORK

This section is divided into two parts. First, we refer to some related work dealing with pedestrian detection or, more generally, object detection in LiDAR data. The second part addresses image-based approaches for pose estimation.

2.1 Pedestrian detection in LiDAR data

Many methods for the detection of pedestrians or, more generally, the detection of specific objects in LiDAR data use a two step approach. First, segments of the processed data are extracted which are supposed to contain individual objects. To achieve this, methods can be used which collate contiguous regions in the processed data, such as *region growing* (Velizhev et al., 2012) or *DBSCAN* (Asvadi et al., 2017). Then, a classifier is used to determine which type of object each segment contains. A weakness of these approaches is the risk of over- or under-segmentation, i.e. cases where multiple objects are part of a segment or a single object is split into multiple segments. Although some classifiers are able to handle such situations to some extent, it may still limit the performance of such approaches.

Voting-based approaches do not rely on a prior segmentation of the processed data, hence they circumvent the risk of over- or under-segmentation. Such methods perform a feature extraction and afterwards use these features to fill a voting space. Objects to be detected cause maxima in the voting space by locally accumulating multiple matching votes, so they can be detected by searching for such maxima. One way to implement such an approach is to use handcrafted features and a dictionary (Knopp et al., 2010, Knopp et al., 2011). The extracted features are matched to the dictionary which contains the information needed to cast a vote. For example, this information can be the relative position of an object's center. More recently, voting-based approaches which use deep neural networks instead of handcrafted features and a dictionary have been proposed (Qi et al., 2019).

Traditional classifiers used for the object recognition in 3D data often rely on a set of handcrafted features. *Support vector machines* (Navarro-Serment et al., 2010), *bag-of-words* (Behley et al., 2013), or *random forests* (Fukano and Masuda, 2015) are examples of such classifiers. More recently, the use of deep neural networks became common. Such networks automatically learn the features they use, which usually leads to features which are better suited for the task at hand than handcrafted ones.

A difficulty of using neural networks on 3D point clouds is their typically unstructured appearance. While camera images have an inherit regular pixel structure, such a grid structure does not exist for the point clouds resulting from, e.g., mobile LiDAR scanning. This makes it difficult to directly use neural network architectures which rely on such a structure. This concerns, for example, convolutional neural networks (CNN), which are currently used very prominently in image processing. One way to handle this is to convert the 3D point clouds to a structured format like depth images (Asvadi et al., 2017, Socher et al., 2012) or voxel grids (Maturana and Scherer, 2015, Garcia-Garcia et al., 2016, Zhou and Tuzel, 2017).

The conversion of 3D point clouds into a structured format can be problematic. The ideal size for the pixel or voxel of such a representation to a certain degree depends on the data density present in the point cloud. If the data density is too high for the chosen pixel or voxel size, information gets lost since many points get grouped together. If the data density is too low, it is difficult to properly fill the grid since there is not enough data available for every voxel or pixel. Usually the data density of single scans of a LiDAR sensor depends on the distance between the sensor and the captured scene, which varies significantly in mobile LiDAR scanning (MLS) use cases. There are approaches which circumvent such issues by using neural network architectures which do not rely on an organization structure of the processed data. (Liu et al., 2019) presented a continuous convolutional neural network which does not depend on structured data, using a specially designed convolution operator *PConv*. *PointNet* (Qi et al., 2017a), later extended to *PointNet++* (Qi et al., 2017b), is another approach which is also able to directly process unstructured point clouds. Their neural networks learn a symmetric function to extract features from the processed data which can then be used for various tasks like classification or semantic labeling.

In our own work (Borgmann et al., 2020) we use a neural network inspired by PointNet to process local point neighborhoods, i.e. small subsets of the point clouds surrounding a certain center point. The output of the network is used to fill a voting space to detect objects. By using relatively small local point neighborhoods, a less complex neural network can be used that requires less training. The work presented in this paper uses this approach for the detection of pedestrians.

2.2 Image-based pose estimation

A wide variety of approaches exists to perform an estimation of human body poses based on images. They can be categorized in different ways. One is to differentiate between approaches which assume that the processed data only shows a single person (Yang and Ramanan, 2013, Dantone et al., 2013, Ke et al., 2018) and ones which can handle multiple persons at once. In case of multiple persons the task becomes more complicated since body parts have to be detected and grouped together for each individual person. In our approach, we already detect individual persons using the LiDAR sensor data. In theory, we could provide the pose estimation component only with image sections of individual persons, but this becomes problematic if multiple persons are in close proximity to each other. In such cases it is difficult to generate image sections which only show a single person. This is especially the case if the sensor placement on the multi-sensor platform leads to greatly different viewing angles for the different types of sensors. Hence, we preferred a multi-person pose estimation for camera image analysis in our approach as well.

For multi-person pose estimation there exist top-down and bottom-up approaches. Top-down approaches detect persons in the processed data and then perform an estimation of body parts and poses for each detected person individually (He et al., 2017, Fang et al., 2017). Bottom-up approaches on the other hand directly detect the body parts of a person and connect the parts belonging to the same person in a later step of the processing. *OpenPose* (Cao et al., 2017, Cao et al., 2019) is a fast and well performing bottom-up multi-person pose estimation approach. It uses a CNN to jointly determine confidence maps for body parts detection as well as Part Affinity Fields (PAFs). These are 2D vector fields which model the location and orientation of limbs which connect two individual body parts. The PAFs are later used to connect the multiple body parts belonging to the same person using a greedy method. In the work presented in this paper we use OpenPose for the estimation of body poses in image data.

3. METHOD

Our approach has two main components: The detection of pedestrians in MLS point clouds and the estimation of body poses for the detected pedestrians in corresponding image sections from multiple cameras. The idea is that MLS point clouds are well suited for the detection and position estimation of pedestrians in the vicinity of a multi-sensor platform. However, the comparatively low data density provided by LiDAR sensors makes them less suitable for determining certain details such as the pose of different body parts. This can better be done using camera images. The approach generates image sections for each detected pedestrian and processes these sections with a state-of-the-art method for human body pose estimation. The resulting poses, which are initially in 2D image coordinates, are then converted back to the 3D coordinate frame.

The approach is designed to be used with data of a mobile multi-sensor system including one or more LiDAR sensors, which are able to provide georeferenced 3D point clouds. The input for the pedestrian detection method is assumed to be a sequence of single LiDAR scans (e.g., single 360° rotations of LiDAR sensors with a rotating scanner head) which share a common world coordinate frame. One axis of this coordinate frame is aligned with the height axis, and we call this axis the z-axis.

Furthermore, the sensor system is assumed to be equipped with one or more cameras. Their intrinsic parameters (e.g., the camera matrix and distortion coefficients) have to be known as well as the geometric constellation of the sensors, i.e. the extrinsic parameters of the whole sensor system. Hence, coordinate transformations between the different involved coordinates frames are possible. While acquiring data, the trajectory and orientations of the sensor platform are measured, which is typically achieved by using an inertial navigation system (INS). This allows the trajectory and orientations of each imaging sensor to be associated with the data it records. The system's sensors also share a common clock, which puts their recordings into a common temporal context.

The technical realization of such a mobile multi-sensor system, which can fulfill these requirements, is a complex topic and is not addressed in this paper. We have described our own experimental system "MODISSA" in a previous paper (Borgmann et al., 2018). In the following, we first describe the pedestrian detection in MLS point clouds and then the image-based body pose estimation used in our approach.

3.1 Pedestrian detection

In this work we use an approach for LiDAR-based detection of pedestrians which was the result of our earlier work (Borgmann et al., 2020, Borgmann et al., 2019). Figure 1 gives an overview of the processing steps of this approach. It generates small local subsets from the acquired 3D point clouds which we call local point neighborhoods. These neighborhoods are processed in a neural network inspired by *PointNet* (Qi et al., 2017a). The network decides whether the neighborhood is part of an object of interest, which in the context of this paper means whether it is part of a pedestrian. If this is the case, the network also estimates where the center of the object is located in relation to the local point neighborhood. These outputs of the neural network are used in a subsequent voting process. The voting process accumulates the results of multiple processed local point neighborhoods and thus mitigates sporadic incorrect estimates made by the neural network. The output of the voting process are the positions of detected pedestrians. We supplement the detection process with a basic tracking method using a Kalman filter. The tracking method handles occlusions and provides additional information about the movement speed and direction of previously detected pedestrians. Certain important aspects of our approach are described in the following subsections.

3.1.1 Pre-processing and local point neighborhoods Although the approach does not necessarily require any previous data filtering, a ground removal can be used to reduce the overall execution time. The approach detects objects mainly based on their appearance and not their surroundings. Therefore, excluding parts of the surroundings from the further processing can be a benefit in terms of execution time, provided that distinguishing between the object of interest and the background is computationally less demanding than the full processing of the data. This is the case for the data points representing the ground. We use a fast method to estimate the ground level which generates a ground grid based on the height represented by the z-coordinate of the points in a grid cell. A subsequent region-growing based validation is used to decide if a certain grid cell actually contains any ground points. By calculating the distance between a point and the ground grid, it is possible to decide whether the point belongs to the ground.

After the ground removal, local point neighborhoods are generated. A local point neighborhood is defined by a center point

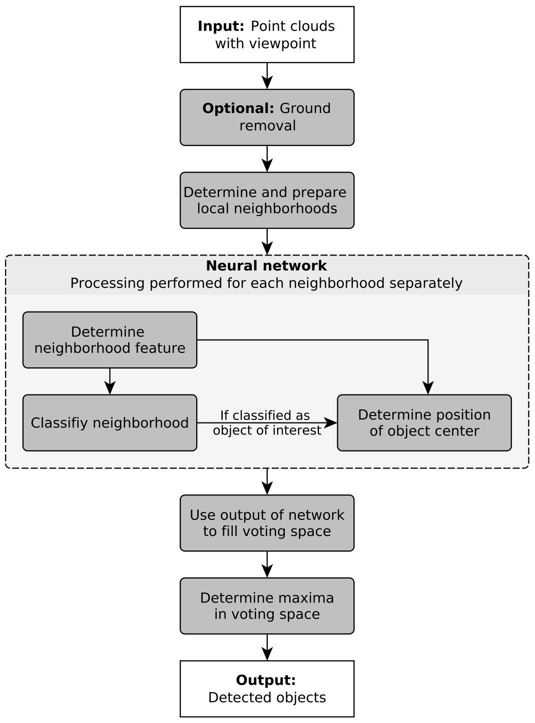

Input: Point clouds with viewpoint
Optional: Ground removal
Determine and prepare local neighborhoods
Neural network
Processing performed for each neighborhood separately
Determine neighborhood feature
Classifiy neighborhood
If classified as object of interest
Determine position of object center
Use output of network to fill voting space
Determine maxima in voting space
Output: Detected objects

Figure 1. Main processing steps of our approach for detection of pedestrians.

and contains all the points in a certain radius around this point. For optimal results, this radius has to fit the data being processed and the specific application. Ideally, the local point neighborhoods should provide an implicit segmentation of the data. Therefore, it is preferable if all points of a local point neighborhood belong only to a small number of objects or only to a single object. In our earlier work, we found that $0.5\,\mathrm{m}$ is a good neighborhood radius for the detection of objects in MLS data of our experimental system, and we use this radius for experiments and results shown in this paper.

Local point neighborhoods have a well defined coordinate frame with the center point as its origin. The orientation of this coordinate frame is defined using the z-axis of the point cloud's coordinate frame as well as the line of sight between the recording sensor and the center point. The approach can generate a local point neighborhood for either every point in the processed cloud or a randomly selected subset of them, depending on a chosen sub-sampling parameter. Again, sub-sampling can significantly improve the execution time and we have demonstrated that low amounts of sub-sampling only have minor influence on the quality of the detection results. Sub-sampling parameters we usually use result in the processing of every third or every fifth point of the original point cloud.

3.1.2 Neural network Figure 2 shows the topology of the neural network used in our approach which follows the design paradigms of PointNet (Qi et al., 2017a). Therefore the network is able to process an unordered set of 3D points. Our local point neighborhoods are significantly smaller than whole point clouds or large point cloud segments normally processed with PointNet. This allows us to use a neural network that is less complex than the original PointNet and therefore easier to train.

The neural network processes local point neighborhoods indi-

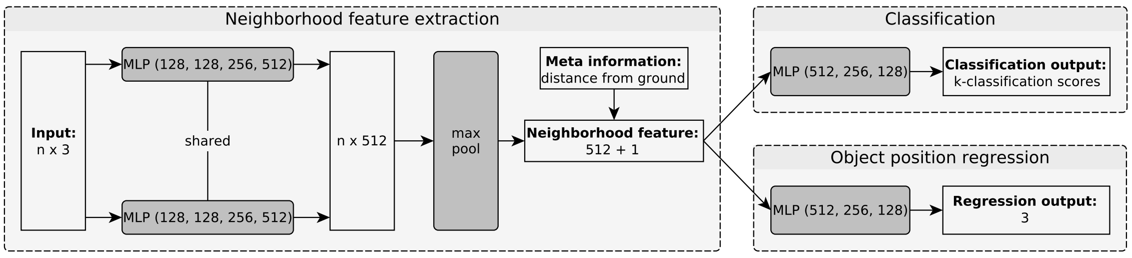

Neighborhood feature extraction
Input: n x 3
MLP (128, 128, 256, 512)
shared
MLP (128, 128, 256, 512)
n x 512
max pool
Meta information: distance from ground
Neighborhood feature: 512 + 1
Classification
MLP (512, 256, 128)
Classification output: k-classification scores
Object position regression
MLP (512, 256, 128)
Regression output: 3

Figure 2. Topology of the neural network. The main input are the n 3D points of a local point neighborhood. The distance from ground to the center point of the processed local point neighborhood is added after the feature extraction to enrich the neighborhood feature. Outputs are classification scores for k classes and estimated 3D coordinates of the object's center. Batch normalization is used for all MLP (*muli-layer perceptron*) layers, except for the layer directly before the output layer. The last batch normalization of each output is followed by a dropout layer with a dropout rate of 0.2.

vidually and has three main parts. The first part of the network extracts a descriptive feature for the processed local point neighborhood. This feature of the size 512 is then enriched by adding the distance between the ground and the center point of the neighborhood as an additional value, giving the resulting feature a size of 513. Adding the distance between ground and local point neighborhood provides the neural network with a basic context information about the location of the neighborhood in the world and has shown to be of benefit for the further processing (Borgmann et al., 2020).

The neural network uses the extracted feature for classification, determining the type of object the neighborhood is a part of. If it is part of an object of interest, a regression estimates the relative position of that object's center. The feature extraction and classification part of the network are trained together. Afterwards, an individual instance of the regression part is trained for each considered object class without further modifying the feature extraction part of the network.

3.1.3 Voting and post-processing The outputs of the neural network for the processed local point neighborhoods are used to generate votes and fill a 3D voting space. During this process, only local point neighborhoods for which the neural network has determined that they are part of an object of interest are considered. Each vote is described by three attributes:

1. Class of the object for which the vote is being cast
2. Center position of the object
3. Weight of the vote

The class and weight of a vote are determined by the classification part of the neural network using the determined object type and its confidence. Instead of using the confidence value directly, the following formula is applied:

$$W_c = \frac{P(c)}{n} \tag{1}$$

where W_c = Resulting weight of a vote for class c
$P(c)$ = Probability of or confidence for class c
n = Amount of points in neighborhood radius

The parameter n is used to approximate the local point density. This should ensure that the overall vote weight in areas of the point cloud with a higher point density is roughly the same as in areas with a lower point density. The position attribute of the votes is determined by the regression part of the neural network, which gets transformed from the neighborhood coordinate frame to the coordinate frame of the point cloud. Hence, the generated votes and the vote space have the same coordinate frame as the processed point cloud.

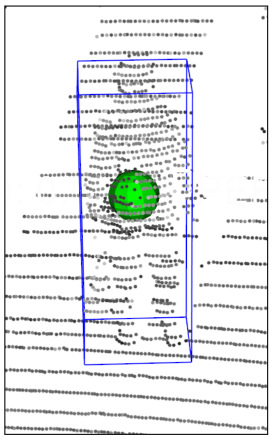
The actual voting process searches for local vote weight maxima. This is done be re-evaluating the weight of each vote by considering the weight of neighboring votes for the same object type. Depending on the distance between the re-evaluated and the neighboring vote, a part of the weight of the neighboring vote is added to the weight of the re-evaluated one. After that a threshold is applied. Votes with a sufficient vote weight are considered as object detections. There can be multiple of such votes for the same object at roughly the same position. Therefore, votes in a close proximity are merged. This results in the output of the object detection as position, type, and score of the detected objects. To additionally determine bounding boxes, all local point neighborhoods whose votes have contributed to the detection of a certain object are considered. The bounding box of an object detection envelopes the center points of all these local point neighborhoods.

3.2 Human body pose estimation

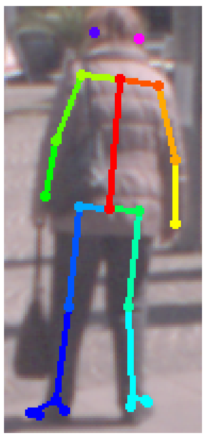
Our approach uses an image-based estimation of human body poses which is performed after the pedestrian detection in 3D point clouds. Figure 3 gives an overview of the processing steps for pose estimation, which are conducted individually for each detected pedestrian. Figure 3a is an example for the output of the pedestrian detection and the input of the pose estimation: A pedestrian detected in an MLS point cloud with a position and bounding box.

For each camera of the multi-sensor system, the camera image is selected whose acquisition time best matches the acquisition time of the 3D point cloud. LiDAR sensors with a rotating scan head do not acquire data at a specific point in time, but scan continuously. It is therefore not possible to perfectly synchronize the recording of a pedestrian in image data and LiDAR data. Small discrepancies between the recording time of the pedestrian in the 3D point cloud and the assigned 2D camera images have to be accepted. Figure 3b gives an example for a camera image assigned to the 3D point cloud.

The 3D corner points of the pedestrian's bounding box are projected onto coordinates of the matched images, utilizing the known geometric constellation of the sensors and knowledge of


(a) Pedestrian detected in a LiDAR point cloud with bounding box.

(b) Full camera image acquired synchronously with the point cloud.


(c) Image section generated based on 3D bounding box, superimposed with the result of the image-based body pose estimation.

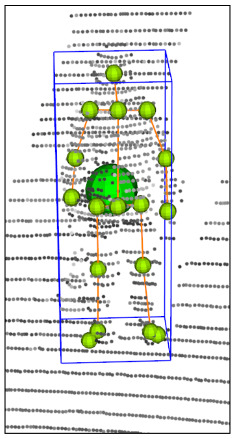

(d) Body pose projected into 3D space.

Figure 3. Processing steps for (a) and (b) image section generation based on detected pedestrians, (c) image based pose estimation, and (d) projection of the poses back to 3D space.

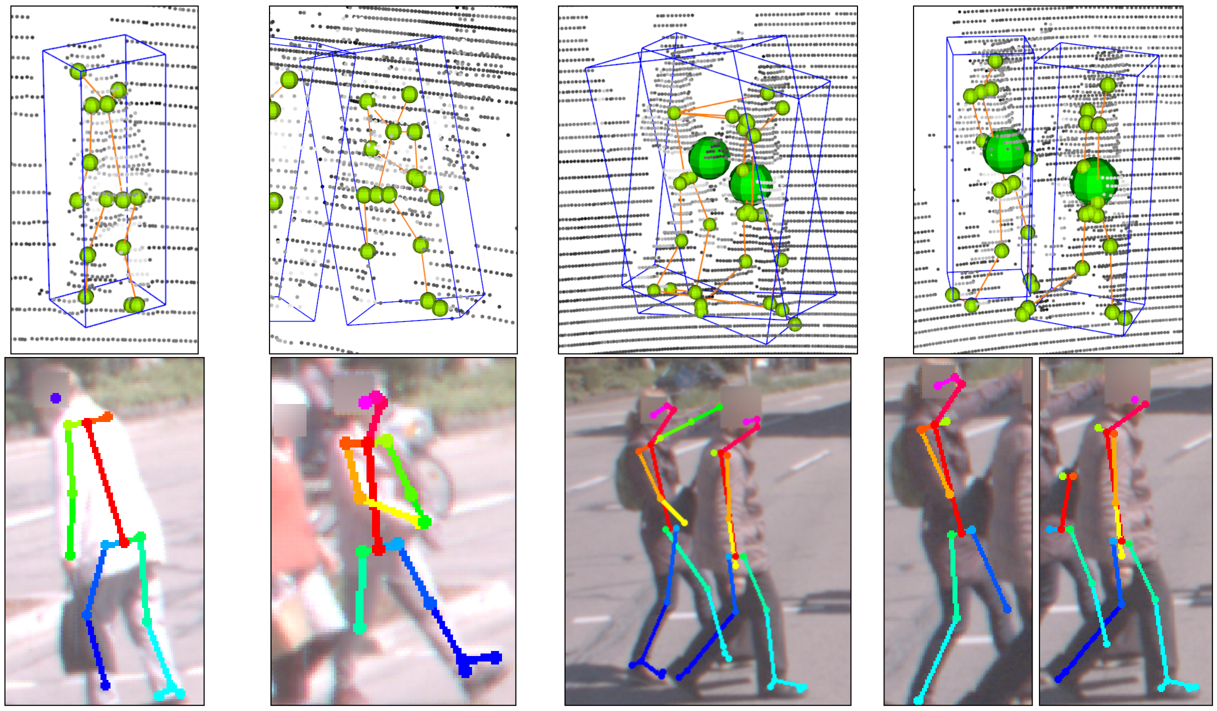
the intrinsic parameters of the cameras. By selecting the corner coordinates which result in the maximum spatial expansion in image coordinates, a 2D bounding box is generated. This does not result in valid image coordinates for every camera, since the different cameras usually have fields-of-view covering only part of the sensor system's surroundings. For the further processing, only camera images with well-overlapping bounding boxes are considered.

The 2D bounding boxes are used to generate image sections which ideally only show the pedestrian to be processed, i.e., whose pose is to be estimated. However, due to different viewing angles of the cameras and the LiDAR-sensors as well as inaccuracies in the determination of the bounding boxes, it is possible that other pedestrians are at least partially visible in the same image section, especially if they are in close proximity to each other. The image sections are processed by a state-of-the-art method for the image-based estimation of human body poses. We use OpenPose (Cao et al., 2019) with the body and foot model for this purpose, but also other equivalent methods can be used in a similar way. Figure 3c shows an image section superimposed with the result of the pose estimation.

OpenPose provides multiple keypoints for the head and the feet of a person, which are more than we need. Therefore, we opted to merge the multiple head keypoints (nose, eyes, ears) to a single combined head keypoint by using their average image coordinate. We did the same for both feet (big toe, small toe, heel).

Based on the intrinsic camera parameters and the known geometric constellation of the sensors, the body keypoints in 2D image coordinates are assigned to 3D rays which originate at the position of the corresponding camera in 3D space at the time the image was acquired. Based on these rays, 3D coordinates are determined for the body pose keypoints by using the 3D points of the point cloud which supported the detection of the processed pedestrian. This means that center points of local point neighborhoods that casted votes for this pedestrian (see Section 3.1) are considered, and 3D pose keypoints are found as the positions on the 3D rays which are closest to one of these center points of the relevant local point neighborhoods.

As mentioned earlier, it is possible that other pedestrians are partially visible in the generated image sections. When processing such an image section, there are pose estimation results for more than one visible pedestrian. To decide which result applies to the correct pedestrian, the one is preferred whose body keypoints in 3D space are on average closer to the supporting points. It is also possible that a specific pedestrian is visible in



(a) A correct result for a single pedestrian.

(b) Multiple pedestrians in proximity to each other with correct results.

(c) Multiple pedestrians in close proximity to each other. The pedestrian detection method detected both, but generated overlapping bounding boxes. That caused too large image sections and errors when projecting the pose into 3D space. The positions resulting from the pedestrian detection are shown as dark green dots.

(d) The same situation as before, but using different parameters for the pedestrian detection, resulting in better bounding boxes but slightly decreasing the performance of the pedestrian detection itself. The positions resulting from the pedestrian detection are shown as dark green dots.

Figure 4. Exemplary results. Pedestrians detected in a 3D point cloud with bounding box and generated image sections superimposed with body pose estimation results. Body pose keypoints have been projected back into the point cloud (light green dots).

the images of multiple cameras. This leads to multiple pose estimation results for the same pedestrian but from different camera images. In such cases, for each type of body keypoint the one with the higher confidence during pose estimation is preferred. With these two heuristics, Figure 3d shows results for different body pose keypoints in 3D coordinates.

4. RESULTS

We conducted experiments with the main focus on the combination of LiDAR sensors and cameras. We already evaluated the pedestrian detection component of our approach in an earlier work (Borgmann et al., 2020). The performance of the body pose estimation method we apply (OpenPose) has already been analyzed by the authors of the original paper (Cao et al., 2019).

For our experiments we used data recorded with a multi-sensor vehicle which we have presented in our earlier work (Borgmann et al., 2018). During our experiments, only one of the vehicle's Velodyne HDL-64E LiDAR sensors was geometrically configured in a way to capture the complete 360° surroundings of the vehicle. We used data of that sensor for the pedestrian detection. During the experiments, the LiDAR sensor was operating with 10 rotations per second, hence a single 360° scan took place in a time period of 0.1 s. Such a scan contains about 130.000 measurements on average, distributed over 64 scan lines. The vehicle also has a panoramic camera setup consisting of two global-shutter RGB video cameras on each corner of the vehicle's roof. All eight cameras combined are able to cover the complete surroundings of the vehicle. We used images from these cameras for the body pose estimation. For the experiments, they were triggered to a frame rate of 10 images per second. The vehicle is equipped with an INS which is used to georeference all acquired data and, in particular, to generate the motion-corrected 3D LiDAR point clouds.

For the experiments we analyzed data which were recorded in an urban street environment. There are multiple pedestrians present either walking along a sidewalk or crossing the road at an intersection. In order to train the neural network for the pedestrian detection, additional short data sequences have been used, most of them also recorded in an urban environment but in a different area than the data used for the evaluation. In total 1300 labeled point clouds have been used for the training and additionally 226 for the validation of the training progress. For the body pose estimation, the body and foot model provided by the developers of OpenPose has been used [1].

[1] https://github.com/CMU-Perceptual-Computing-Lab/openpose

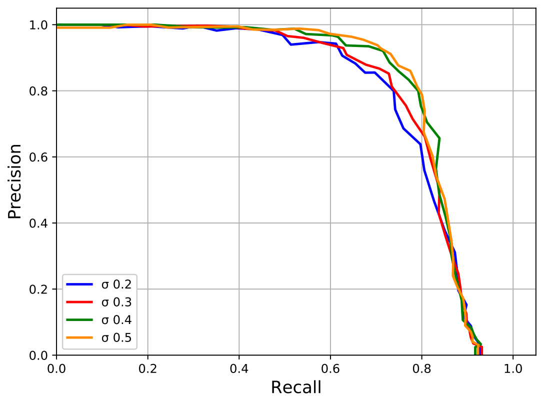


Figure 5. Precision-recall curves for pedestrian detection with different settings of the parameter σ

For legal and data protection reasons, we used image data in which the pedestrian's faces were blurred promptly after data recording. This limits the capability of OpenPose in regards to detecting keypoints on the head. But since our experiments are not focused on the performance of OpenPose, we accept this imposed weakness of our data.

We did a manual assessment of the quality of the results to find weaknesses in our approach. Figure 4 shows some exemplary results. Our approach works reasonably well in situations in which pedestrians have sufficient distance to each other, as shown by Figure 4a, and in many cases in which they are closer to each other, as shown by Figure 4b. A general weakness is that the body keypoints, when transferred back to the 3D point cloud data, always end up on the surface of the pedestrian. This is caused by intersecting the 3D rays with the LiDAR data, which represent the surface of the pedestrian instead of his or her interior. Another possible problem can be caused by the short time gap between the data recordings of the LiDAR sensor and the cameras. As explained earlier (see Section 3.2), due to the characteristics of scanning LiDAR sensors it is not possible to fully synchronize the capturing of a specific position in the world by the LiDAR sensor and the cameras. The movement of the sensor system itself is not an issue, since this movement is taken into account by the INS. However, movements of the pedestrians themselves can be problematic, if these movements are fast compared to the scanning speed of the LiDAR sensors and the frame rate of the cameras. This is especially the case as pedestrians move their limbs, changing their appearance as they move. This can cause inaccuracies when projecting the 2D body pose keypoints into 3D space. A technical solution for this problem is to increase the frame rate of the cameras, which minimizes the potential time gap between a given LiDAR scan and the best matching camera image.

Figure 4c shows a problem which can occur when multiple pedestrians are in close proximity to each other. While still being able to detect these persons individually, our pedestrian detection method may generate bounding boxes which at least partially include both pedestrians. This causes two problems for the body pose estimation: One is that the generated image sections are too large. The other is that while determining the 3D coordinates of the body pose keypoints, 3D points of both pedestrians are considered which can cause wrong results. The bounding boxes are generated in a way that they include all the 3D points whose local point neighborhoods and subsequent votes supported the detection of the pedestrian. As explained in Section 3.1.3, during the voting process the weight of each vote is re-evaluated based on its neighboring votes for the same type of object. These neighboring votes are considered to be in support of the one currently re-evaluated. This means the radius up to which votes are considered to be neighbors has an influence on the bounding box generation. This radius is controlled by a parameter σ. If the radius is too large, it could include votes which actually belong to another pedestrian close by, which is what happened in the given example. A solution for this issue is to reduce the value of the parameter σ, which results in more accurate bounding boxes and solves the subsequent problems as shown by Figure 4d. But this solution has a small negative impact on the detection performance of the pedestrian detection itself, as is shown by Figure 5. This figure presents the result of an experiment where we have compared the detection performance for different values of σ and it shows the slight decrease of the overall detection performance if this parameter gets smaller.

5. CONCLUSION

We presented an approach which combines multiple sensor types of a vehicle-based multi-sensor system and uses them to gather information about pedestrians in the vicinity of the vehicle. The LiDAR sensors are used for the detection and tracking of pedestrians while the cameras are used for a detailed body pose estimation.

The approach is able to combine the specific advantages of LiDAR sensors and cameras to provide more comprehensive information about pedestrians in the vehicle's vicinity. It aims to minimize computational overhead by performing image analysis only for image sections which are regions of interest and actually contain pedestrians. This lowers the overall execution time, especially if multiple cameras are used. By using the knowledge which 3D points of the point cloud belong to a particular detected pedestrian, the approach is able to determine 3D coordinates for the 2D results of the pose estimation. Although some weaknesses exist, the approach has shown promising results for its intended use cases.

For the future, we plan to use a camera mounted on a pan-tilt unit that has a narrow field-of-view and provides a locally higher level of detail. By pointing such a camera in the direction of LiDAR-based pedestrian detections, we expect to be able to acquire and process high-resolution images of individual detected pedestrians.

REFERENCES

Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., Nunes, U. J., 2017. DepthCN: Vehicle detection using 3D-LIDAR and ConvNet. *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–6.

Behley, J., Steinhage, V., Cremers, A. B., 2013. Laser-based Segment Classification Using a Mixture of Bag-of-Words. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4195–4200.

Borgmann, B., Hebel, M., Arens, M., Stilla, U., 2019. Using neural networks to detect objects in MLS point clouds based on local point neighborhoods. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W7, 17–24.

Borgmann, B., Hebel, M., Arens, M., Stilla, U., 2020. Pedestrian detection and tracking in sparse MLS point clouds using a neural network and voting-based approach. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2020, 187–194.

Borgmann, B., Schatz, V., Kieritz, H., Scherer-Klöckling, C., Hebel, M., Arens, M., 2018. Data processing and recording using a versatile multi-sensor vehicle. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1, 21–28.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., Sheikh, Y. A., 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cao, Z., Simon, T., Wei, S.-E., Sheikh, Y., 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *CVPR*.

Dantone, M., Gall, J., Leistner, C., Van Gool, L., 2013. Human Pose Estimation Using Body Parts Dependent Joint Regressors. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 3041–3048.

Fang, H., Xie, S., Tai, Y., Lu, C., 2017. RMPE: Regional multi-person pose estimation. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2353–2362.

Fukano, K., Masuda, H., 2015. Detection and Classification of Pole-like Objects from Mobile Mapping Data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W5, 57–64.

Garcia-Garcia, A., Gomez-Donoso, F., Garcia-Rodriguez, J., Orts-Escolano, S., Cazorla, M., Azorin-Lopez, J., 2016. PointNet: A 3D Convolutional Neural Network for Real-Time Object Class Recognition. *2016 International Joint Conference on Neural Networks (IJCNN)*, 1578–1584.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988.

Ke, L., Chang, M.-C., Qi, H., Lyu, S., 2018. Multi-scale structure-aware network for human pose estimation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 713–728.

Knopp, J., Prasad, M., Gool, L. V., 2011. Scene Cut: Class-Specific Object Detection and Segmentation in 3D Scenes. *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 180–187.

Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L., 2010. Hough transform and 3D SURF for robust three dimensional classification. *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, Springer-Verlag, Berlin, Heidelberg, 589–602.

Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., Pan, C., 2019. DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing. *IEEE International Conference on Computer Vision (ICCV)*, 5239–5248.

Maturana, D., Scherer, S., 2015. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928.

Navarro-Serment, L. E., Mertz, C., Hebert, M., 2010. Pedestrian Detection and Tracking Using Three-dimensional LADAR Data. *The International Journal of Robotics Research*, 29(12), 1516-1528.

Qi, C. R., Litany, O., He, K., Guibas, L. J., 2019. Deep Hough Voting for 3D Object Detection in Point Clouds. *Proceedings of the IEEE International Conference on Computer Vision*.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 77–85.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds), *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., 5099–5108.

Socher, R., Huval, B., Bath, B., Manning, C. D., Ng, A. Y., 2012. Convolutional-Recursive Deep Learning for 3D Object Classification. *Advances in Neural Information Processing Systems*, 656–664.

Velizhev, A., Shapovalov, R., Schindler, K., 2012. Implicit shape models for object detection in 3D point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-3, 179–184.

Yang, Y., Ramanan, D., 2013. Articulated Human Detection with Flexible Mixtures of Parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12), 2878-2890.

Zhou, Y., Tuzel, O., 2017. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection.