High-Speed Recognition of Pedestrians out of Blind Spot with Pre-detection of Potentially Dangerous Regions

Jiacheng Zhou[1], Masahiro Hirano[2] and Yuji Yamakawa[3]

***Abstract*— This paper presents a novel approach for the early detection of pedestrians crossing out of blind spots. We used a monocular camera mounted at the front of a vehicle and developed an algorithm for detecting blind spots caused by obstacles blocking the view of the driver, based on the depth information obtained from a monocular depth estimation before the vehicle approaches a potentially dangerous region. Unlike other studies in which pedestrians were searched for over the entire image, our blind spot pre-detection method allows for a considerably smaller search area for pedestrian detection, which can significantly reduce the time latency of detecting unexpected pedestrians. We evaluated the proposed method on test videos with complex pedestrian and street scenes. The experimental results revealed that the performance of the method in detecting blind spots and sudden appearance of pedestrians is good, and the reaction latency for such emergency situations is significantly reduced by applying the pre-detection method. Our method can be applied in advanced driver-assistance systems and reduce accident rate caused by sudden pedestrian crossing.**

I. Introduction

Traffic safety has always been an important issue in people's daily lives. However, because of the popularity of smart phones, numerous people tend to be immersed in the virtual world even when they are walking on the streets, significantly increasing the danger of being hit by approaching vehicles. This would be even more dangerous if pedestrians, especially children, suddenly ran out of blind spots, such as large buses and walls around crossings. Owing to the slow reaction of humans, drivers can rarely apply brakes or maneuver their vehicles in time. Every millisecond is important considering the high speed of a moving vehicle [10]. Computer vision can help drivers detect pedestrians suddenly appearing out of a blind spot at the earliest moment, as depicted by the man in orange in Fig. 1.

Two main types of methods are available for pedestrian detection in autonomous driving: traditional methods such as handcrafted features combined with a sliding-window framework and deep learning methods such as convolutional neural networks (CNNs) [2]. Most methods for detecting sudden pedestrian crossing (SPC) involve a pedestrian detector on the main part of the monocular image [3] or obtaining stereo information by using additional devices such as depth cameras, binocular cameras, and laser scanners [4]. However, searching for unnecessary regions in an image can be time-consuming and require additional costs.

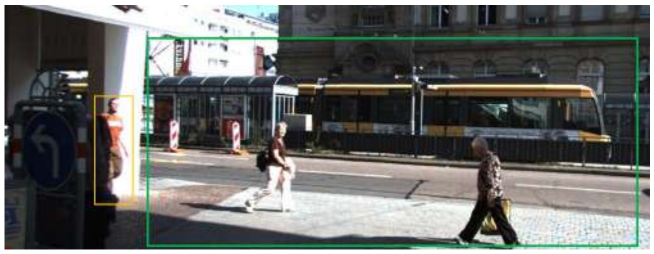


Fig. 1. Different regions in the same image. Image region can have sudden pedestrians (orange), and region can be neglected (green)

In this paper, we introduce a method for dealing with the SPC problem; it requires only a monocular camera and can detect pedestrians suddenly appearing within a potentially dangerous region. Our method can be divided into three stages: blind spot pre-detection, sudden pedestrian detection, and danger-level analysis. The first stage is implemented when a vehicle approaches the potentially dangerous region. To localize the obstacles that might block a driver's view along the two sides of a vehicle, we propose a depth-gradient-feature-based algorithm that can detect the edge of the obstacle through a rough pixel-wise depth map obtained from the monocular depth estimation method. Applying this pre-detection stage has two benefits. First, it enables faster pedestrian detection at the target blind spot than conventional methods, which require a full image analysis. The second benefit is explained in combination with the second stage.

The second stage is pedestrian localization, where a deep-learning-based method is used. However, fast deep-learning-based object detection methods have limitations in detecting small-sized or half-blocked people, especially when the region of interest (ROI) is large and contains several objects [1]. Therefore, we propose to restrict the searching area to a region where pedestrians may suddenly appear and neglect regions that can be clearly seen by the driver, such as the green region in Fig. 1. Our blind spot pre-detection method can well-serve this purpose.

The third stage is motion detection and danger analysis. We first calculate the average translation of the detected pedestrian through optical flow. We then analyze whether the pedestrian is at a risk of being hit by the vehicle.

The main contribution of our study is that we propose a new method that can help vehicles quickly respond to sudden

*This work was not supported by any organization

[1]Jiacheng Zhou is with the Department of Mechanical Engineering, School of Engineering, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan jiacheng@iis.u-tokyo.ac.jp

[2]Masahiro Hirano is with the Department of Mechanical and Biofunctional Systems, Institute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan mhirano@iis.u-tokyo.ac.jp

[3]Yuji Yamakawa is with Interfaculty Initiative in Information Studies, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan y-ymkw@iis.u-tokyo.ac.jp

pedestrian appearances out of blind spots. The proposed algorithm can detect the edges of obstacles blocking the view of a driver using only rough depth information obtained from monocular depth estimation, which can significantly reduce the reaction time. Experiments are conducted to examine the performance of our blind spot pre-detection method in terms of the reaction time and compare it with that of the direct application of the pedestrian detection method.

II. Related Work

At present, researchers are paying considerable attention to pedestrian detection. However, despite its significance, the SPC problem has been investigated in very few studies. In one of the earliest studies on this problem, Xu et al. [3] proposed a three-level coarse-to-fine framework based on sliding windows that could detect pedestrians suddenly appearing at the edge of an image. To reduce the pedestrian detection region and boost reaction speed, they combined a sparse sliding window with a motion filter based on a local binary pattern to select windows with significant motion at the local level. Although this motion filter cuts the processing region, it still regards all moving pedestrians equally. We consider the environmental information and employ computing resources on only the blind spot regions. Hence, our method requires a less amount of computations for pedestrian detection and danger level analysis.

The concept of detecting a potentially dangerous region in advance was proposed by Broggi et al. [4], whose approach only searches for pedestrians in areas where obstacles can block the view of a driver. However, they used a laser scanner to reconstruct the environment and find the blind spots. This can significantly increase the implementation and maintenance costs. Additional work on sensor fusion is also necessary in this regard. Moreover, they needed a few laser scanner rotations for corrections to achieve a higher depth accuracy, thereby limiting the processing speed. Our method enables the usage of a relatively low-cost monocular camera instead of laser scanners. Moreover, the reaction speed of the proposed method can be higher.

Jeong et al. [5] also studied the SPC problem by mounting a far-infrared camera on top of a vehicle and using the temperature for area segmentation and a reference line to reduce the detection region. Then, they combined a cascade random forest with low-dimensional Haar-like features and oriented center-symmetric local binary patterns to detect multiple pedestrians. However, the pedestrian searching area accounted for more than half of the entire image and considered pedestrians in front of the vehicle. By contrast, our blind spot pre-detection method works harmonically with the current system for the SPC problem and restricts the ROI to a substantially smaller size.

III. Sudden Pedestrian Detection out of Blind Spots

Our method is illustrated by the flowchart in Fig. 2. The first stage is blind spot detection, which involves depth estimation and obstacle edge detection. Obstacle edge detection involves two modes: depth estimation mode (DEM) and blind spot prediction mode (BSPM), which are discussed in detail in subsequent sections. The second stage is pedestrian detection, in which pedestrian detectors such as YOLO are applied within the cropped blind spot region. If any pedestrian is detected, the third stage of motion detection and danger level analysis is applied, and the pedestrian's velocity is calculated through optical flow.

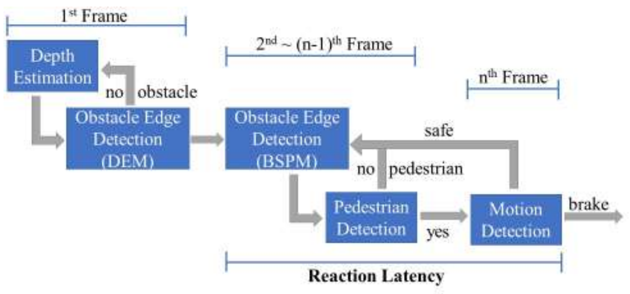


Fig. 2. Flowchart of the method

A. Blind Spot Detection

The first stage of our method is used to detect obstacles (such as walls and buses) that block a driver's view alongside a vehicle. The corresponding region, which can have the sudden appearance of a pedestrian, is then defined as a rectangle area next to the edge of the obstacle. This stage is divided into two modes based on different times: DEM and BSPM. When the edge of the obstacle is far away from the vehicle, the DEM is implemented to search for the potential edge of the obstacle in a relatively time-costly manner. When the edge is detected and the vehicle comes in close proximity to the potentially dangerous region, the DEM is stopped and the BSPM is started to predict the edge position according to speed of the vehicle; this has negligible time cost. Our method is used to avoid collisions with suddenly appearing pedestrians. Therefore, detecting pedestrians coming out of a blind spot when the vehicle is still far away from potentially dangerous region is unnecessary. Thus, only the suddenly appearing pedestrians detected in the BSPM need further attention. In the DEM, we first use monocular depth estimation to estimate the pixel-wise depth map of the current frame and then develop an algorithm for identifying any sudden depth decrease to locate the edges of obstacles.

1) Monocular Depth Estimation: For depth estimation, we use Monodepth2 proposed by Godard et al. [6]. They applied self-supervised learning and can train on monocular sequential images to obtain an inference model for estimating pixel-wise depth information from a single image. The performance is comparable to that of a supervised learning model. The inference output is a pixel-wise matrix of the reciprocal of the normalized depth value.

In this study, we train our Monodepth2 model based on the KITTI dataset [8]. As the amount of data on pedestrians walking out of blind spots is limited, we split the dataset and include images with more pedestrians and obstacles into

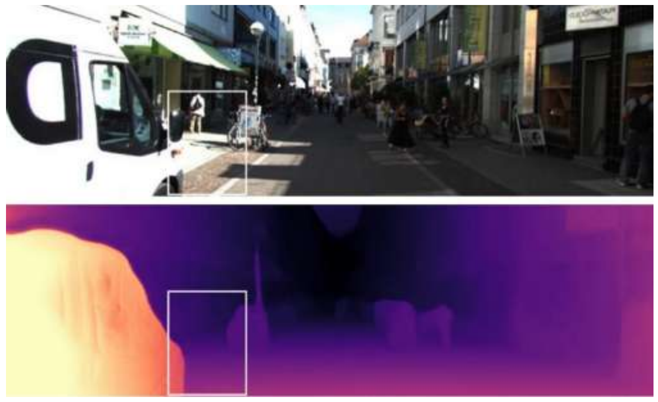


Fig. 3. An example of Monodepth2 inference and detected blind spot

the test set for the evaluation. Fig. 3 displays a depth image example by Monodepth2. The first image is the input and the second one is the depth map output.

2) Blind Spot Detection using Depth Maps: In this study, obstacles are defined as objects that are big enough to block a driver's view alongside the vehicle, as depicted by the van on the left side in Fig. 3. The white box is detected as a blind spot region in the DEM and excludes most parts of the image, thus significantly boosting the reaction speed.

We develop an algorithm to find the edge of the obstacle based on depth maps. For illustration, the following algorithm considers only the left part of the image. When a vehicle is moving and the edge of the obstacle is not yet detected, the DEM is applied on each frame. When the edge of any obstacle is detected, the blind spot is defined as a white box in Fig. 3. Then, if blind spots are detected through the DEM within more than two sequential frames or the distance between the vehicle and the obstacle reaches the threshold value, in the following few frames, the DEM is stopped and the BSPM is started to predict the blind spot location, which is displayed as a red box in Fig. 5. Details of the DEM and BSPM are discussed in the following sections.

DEM: Three steps are performed in the DEM to find an obstacle edge. The first step is checking the depth gradient along the road direction and finding the sudden depth drop position. As the red line in Fig. 4 shows, we select an inclined line segment starting from $(0, 3/4H)$ to $(3/10W, 3/5H)$ as the initial reference line, which has approximately the same direction as the road. The gradient features of equally divided points every five pixels on the reference line are checked to find the sudden depth drop position. For example, the depth vector along the reference line can be denoted as:

$$dv = [d_1, d_2, \cdots, d_n]$$

Each d_i represents the depth information of the pixel selected from the reference line. Then, the depth gradient vector is calculated as:

$$dgv = \left[\frac{d_2 - d_1}{5}, \cdots, \frac{d_n - d_{n-1}}{5}\right]$$

The peak value within the depth gradient vector is selected, and the sudden drop point on the reference line can be obtained by using the indices of the peak values; the sudden drop point is denoted as the red point in Fig. 4. Its x coordinate is denoted as x_0, and the y coordinate, y_0, can be calculated using the linear relation of the reference line.

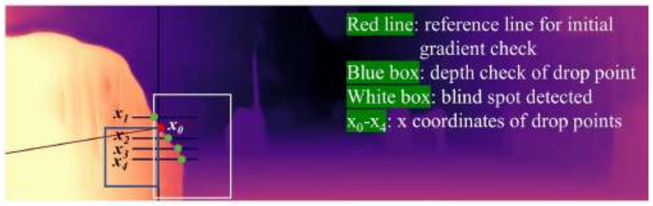
The second step is checking the depth to determine whether an obstacle big and close enough to the vehicle exists at the drop point. A rectangular box is selected to the left of the drop point, as shown in blue in Fig. 4. Pixels are sparsely picked within the box, and their depths are checked. If more than 90% of the selected pixels within the box have a depth smaller than the threshold, the depth check is passed and an obstacle that can block the view of the driver is considered to exist. This blue box determines the smallest size of an obstacle that can block the view of the driver; the size can be modified depending on different camera settings. We use 100×80 pixels for the KITTI dataset [8].

When the second step is passed, the third step ensures that an obstacle edge exists at this position rather than a random depth drop along the reference line. In this step, other horizontal line segments are selected, and gradient checks are applied to each of them, as shown by the four horizontal dark blue line segments in Fig. 4. The drop points on these four line segments are represented by green, and their x coordinates are denoted as x_1, x_2, x_3, and x_4. If more than two drop points exist, their values are placed into set S and the final x coordinate of the sudden depth drop is calculated by applying the following equation:

$$x_d = \frac{1}{n}\sum_{i=1}^{n}(x_i)$$

x_i is an element in S and n is the size of S. If no other drop points are detected, we consider that no obstacle edge exists close enough to the vehicle and skip this frame.

After the average position of the detected drop points is obtained, the blind spot region is determined as a 192×160 white rectangle with the top left point as $(x_d - 20, y_t)$ (shown in Fig. 4), where y_t is based on the image size and camera location. This size is chosen because the YOLOv5 [7] inference model allows sizes that are divisible by only 32. The relatively larger width with respect to pedestrians is designed to compensate for drop point detection errors.


Red line: reference line for initial gradient check
Blue box: depth check of drop point
White box: blind spot detected
x_0-x_4: x coordinates of drop points
x_1
x_0
x_2
x_3
x_4
Fig. 4. Blind spot detection algorithm

BSPM: The BSPM is applied immediately after the DEM and is considerably faster. If any blind spot is detected in more than two consecutive frames or the latest position of the blind spot is sufficiently close to the image edge, the vehicle is considered to be approaching the place where SPC may occur. In this case, the DEM is ended and the BSPM is started. In the BSPM, the position tracking of a blind spot region is designed to be based on the vehicle speed and the camera parameters. However, in our cases, because real-time velocity information cannot be obtained from the KITTI dataset [8], we use the highest speed in the previously detected blind spot from the DEM, which is the largest pixel translation per frame of the white box in the x direction, as the speed of the blind spot in the BSPM. The blind spot location can be predicted with negligible time cost without applying Monodepth2 inference and the depth-gradient-feature-based algorithm. The BSPM and DEM can be combined to rapidly identify the blind spot region as the vehicle approaches a potentially dangerous location.

B. *Pedestrian Detection*

After the blind spot region is detected, the pedestrian detection method is applied to it. We apply YOLOv5 [7], which is an open-source object detection architecture that can rapidly and accurately detect objects, including pedestrians. Moreover, YOLOv5 [7] is considered to be more efficient and faster than sliding window methods, such as the method used by Xu et al. [3]. As the inference speed and accuracy of the YOLOv5 model are sensitive to the input image, the time needed for pedestrian detection in the blind spot region is much less than that for the entire image. Pedestrian detection examples where YOLOv5 [7] is applied to the predicted blind spots are shown in Fig. 5. The pedestrian is accurately detected and bound within the green/red box. The details of time consumption and a comparison are provided in the subsequent sections.

C. *Motion Detection and Danger Level Analysis*

The third stage is employed to analyze the motion of the detected pedestrian within the blind spot. If the pedestrian is moving toward the vehicle at above a certain speed or becoming extremely close, they will be detected as “dangerous.” To determine the speed of the detected pedestrian along the x direction, optical flow [11] is applied between two sequential frames. If a high-speed monocular camera is used, time latency can be reduced significantly. Optical flow functions when a pedestrian is detected. Corner points are selected within the green box region of the first frame. Then, in the next frame, the nearby region of the previous pedestrian is checked using the Lucas–Kanade (LK)-based optical flow [11] to find the corresponding corner points. The average pixel translation of the pedestrian within the two frames can then be obtained, based on which the average velocity can be calculated by applying the following equation, where x_t is the average pixel translation (pixel) along the x direction and t is the time interval (second) between continuous frames.

$$v = x_t/t$$

If the pedestrian moves toward the center of the image and the average velocity exceeds the threshold, the vehicle should perform a collision avoidance maneuver. Fig. 5 shows examples of motion detection. The small red box indicates that a pedestrian is detected as walking toward the potentially dangerous area and is in risk of being hit by the vehicle. The green box indicates that the pedestrian is walking away from the vehicle; therefore, the vehicle does not need to apply brakes.

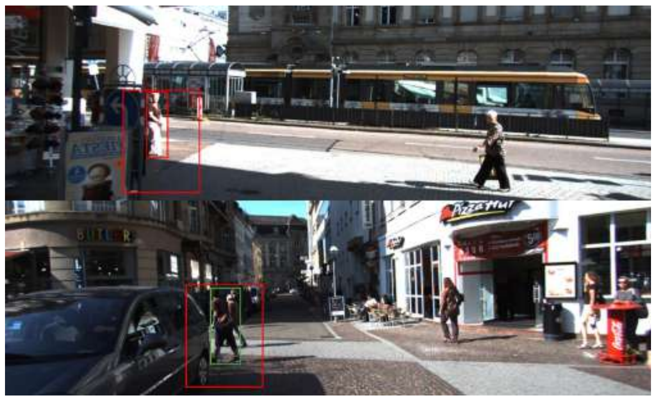


Fig. 5. Examples of pedestrian detection and danger level analysis

IV. Evaluation

A. *Evaluation Setup*

We used a computer with the following configuration: Intel® Core™ i7-6950X CPU @ 3.00 GHz × 20 and TITAN X (Pascal)/PCIe/SSE2 GPU. The data used to train the Monodepth2 model and for test purposes were acquired from the monocular image parts of the KITTI2011 dataset [8] and included many different situations, such as urban areas and highways. The monocular camera was mounted on top of the vehicle, and the frame rate was 10 fps. For pedestrian detection, we used the YOLOv5s model pretrained on the COCO dataset [9] as the inference model. To increase the processing speed to the maximum, we applied TensorRT [12] to the YOLOv5 network. TensorRT is a software development kit created by NVIDIA for high-performance deep learning inferences. Finally, in all the tests of the proposed method, only the left part of the image was considered. Therefore, the overall reaction times for pedestrian detection and motion detection should be doubled if both sides are to be detected.

B. *Evaluation Criteria*

In Fig. 6, when the pedestrian appears in the first frame and is detected in the second frame, her translation is calculated through optical flow in the third frame. Our time latency for detecting the pedestrian can be calculated as $t = t_1 + t_2 + t_3 + t_4$. Here, t_1 is the time for blind spot detection. This stage is in either the DEM or the BSPM based on different times. As our aim is to evaluate the time latency under emergency situations, when the DEM is stopped and the BSPM is employed, t_1 is the time cost of the BSPM, which is negligible and regarded as zero.

The second term, t_2, represents time from the pedestrian appears to the pedestrian is detected. In this study, the interval for each frame is 100 ms. Thus, t_2 is calculated as $t_2 = n \cdot 100$ ms, where n is the number of frames needed to successfully detect the sudden pedestrian appearing from behind an obstacle.

The third term, t_3, represents the larger one between frame interval and $t_d + t_c$, where t_d is the time for pedestrian detection and t_c is time for corners selection among the detected pedestrian image region in the same frame. However, since time interval for KITTI dataset [8] (100ms) is much larger and totally cover $t_d + t_c$, t_3 is 100ms. t_d highly depends on the detection region. We also show t_d based on YOLOv5 [7] with TensorRT over images of different sizes.

The fourth term, t_4, is the time consumption for detecting corners' corresponding points in the next frame. The sum of t_c and t_4 is the total time consumption of LK optical flow. To illustrate that the proposed blind spot pre-detection method helps increase the reaction speed, we compare the time latency of methods with and without pre-detection.

C. Results

1) Time Consumption of Each Part: After the test on the KITTI dataset [8], the average values of t_d are obtained for the YOLOv5s inference model of different input sizes, as shown in Table. 1. The time for pedestrian detection increases as the detection region becomes larger. The input size of the YOLOv5 model is chosen because the image size of the KITTI dataset [8] in this study is 375×1242 ($height \times width$) and the size of the cropped blind spot region is 192×160. If the width and height are divisible by 32, YOLOv5 will not apply image resizing and, thus, save time. For optical flow part, t_c and t_4 has an average value of 3.913 ms and 1.785 ms. n in t_2 is 1 for KITTI dataset [8]. Also, the average time consumption for the DEM is 56.785 ms per frame; however, the DEM only functions before the vehicle approaches, and thus, the DEM part need not be included in the emergency time latency.

TABLE I
AVERAGE TIME CONSUMPTION OF DIFFERENT YOLOV5S MODEL SIZES

Input size for YOLO	Inference time (t_2)
YOLOv5s 192×160	4.053 ms
YOLOv5s 352×352	6.139 ms
YOLOv5s 352×1216	10.196 ms

2) Comparison of Latencies with and without Blind Spot Pre-detection: To demonstrate the benefits of the proposed blind spot pre-detection method, we compare the method with the direct application of YOLOv5 to the entire image. Three frames in Fig. 6(a) present one of the test results. The pedestrian within the blind spot (large red box) is detected in the second frame (green box), even if she is partially blocked. Then, her motion is calculated based on the second and third frames through optical flow. As the motion is toward the image center, she is considered as "dangerous." Consequently, the green pedestrian box becomes red to denote "alert." Fig. 6(b) shows the direct application of YOLO (352×1216) to the entire image; this misses the pedestrian in the second frame and works only after the pedestrian completely appears in the third frame.

The results indicate that the proposed pre-detection method can help detect an unexpected pedestrian at an earlier stage with a lower time latency. In this case, the time latency for sudden pedestrian detection is reduced by 100 ms, which is one frame time interval. The latency decrease is significant because the frame rate of this dataset is small. However, if a high-speed camera more than 100fps is used, t_3 part will be replaced by $t_d + t_c$ and the reaction latency is also reduced by applying smaller YOLOv5 inference image as shown in Table. 1 thanks to the blind spot pre-detection method.

Actually, t_2 is not meaningful for evaluation since the very first frame where pedestrian appears can also be defined when he/she half appear, such as the second frame in Fig. 6. Hence, counting from the second frame, the overall reaction time for this case can be calculated by applying the following equation

$$t = t_3 + t_4 = 100 + 1.785 = 101.785\text{ms}$$

In this case, the reaction latency is relatively large, limited by small frame rate of KITTI dataset [8]. However, if a camera with a higher frame rate is used, the proposed method is expected to perform better in real-time applications for assisting drivers in emergency situations. If 500 fps data is used, as our future work, total time latency can be reduced to less than 10 ms. Also, the speed for the optical flow part can also be increased according to the result obtained by Xu et al. [3]. Furthermore, other cases show that the proposed blind spot pre-detection method can help to significantly reduce the reaction latency for pedestrians detected at an earlier frame. Further evaluations based on self-collected data will be done.

However, the proposed method may fail on some occasions. In the first stage of blind spot detection, monocular depth estimation can fail for some distorted, reflective, and color-saturated regions [6]. In this case, some negative cases of blind spots can be detected as positive. For the second stage, pedestrian detection performance is limited by the performance of the YOLO model. Using different models such as "nano," "s," "m," and "l" will impact the accuracy/latency balance. In the test cases in this study, because of blind spot pre-detection, YOLOv5s detected pedestrians well with no failure cases.

V. CONCLUSION

We proposed a new method to help drivers react more quickly when a pedestrian suddenly appears out of a blind spot by pre-checking potentially dangerous regions. The KITTI dataset [8] was used to test the method, and the experimental results showed that the method is suitable for real-time applications. Even in complex test situations, such as the urban view in Fig. 6(a), the proposed method performs well with a short reaction time. The results also showed that blind-spot pre-detection can help detect sudden pedestrians at an earlier stage than when directly applying pedestrian detection on the entire image. The proposed pre-detection

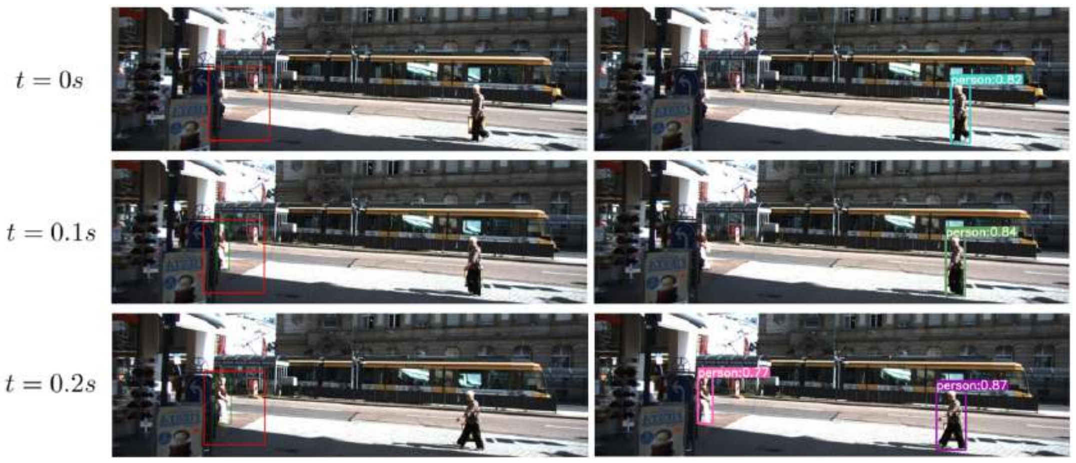


Fig. 6. (a). Test results using blind spot pre-detection method (left). (b). Test results by directly applying YOLO to entire images (right)

method will be beneficial to any type of pedestrian detection methods because of the smaller inference image sizes.

However, this study has some limitations. First, this study is based on the assumption that the vehicle and the obstacle have constant speeds in the BSPM. The vehicle part can be solved by obtaining the real-time vehicle velocity from controller area network or other sensors. However, for the obstacle part, the obstacle can be, for example, a bus moving towards our vehicle, and the speed may change during the BSPM. In this situation, we can start the BSPM later and assume that the speed of the obstacle is constant in the previous few frames, making a prediction failure less likely. Second, this study involved the use of the KITTI dataset [8], whose image frequency is 10 fps. Hence, most of the reaction time was wasted because of this low frequency. Using a monocular camera with a higher frequency will enable the proposed method to have a much smaller latency. Additionally, as the KITTI dataset [8] lacks situations of SPC, appropriate evaluation of the reliability of the proposed method is not feasible owing to such a small number of test scenarios. We will conduct further evaluations by using the road data we collect. However, the effectiveness of our blind spot pre-detection method in boosting the reaction speed in SPC problems is already revealed in this study.

In future work, we will collect our own road data for more emergency situations using high-speed cameras. We can train the Monodepth2 model based on the data collected and evaluate the performance, including accuracy and reaction latency, of the proposed method by using high-speed cameras. Furthermore, the latency from a low camera frame rate, which is the primary latency in the current study, can be reduced significantly. Real-time vehicle speeds will also be applied to the BSPM so that the obstacle edge can be tracked more precisely. This study will also be expanded to detect the sudden appearances of cars and people riding bikes out of blind spots, which necessitate faster reaction times. Finally, we will also consider different driver maneuvers to avoid collisions in different situations.

References

[1] Z. Yang, J. Li and H. Li, "Real-time pedestrian and vehicle detection for autonomous driving," *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 179–184. doi: 10.1109/IVS.2018.8500642.

[2] L. Chen et al., "Deep neural network based vehicle and pedestrian detection for autonomous driving: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, IEEE, June, 2021, pp. 3234–3246. doi: 10.1109/TITS.2020.2993926.

[3] Y. Xu, D. Xu, S. Lin, T. X. Han, X. Cao and X. Li, "Detection of sudden pedestrian crossings for driving assistance systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, IEEE, 2012, pp. 729–739. doi: 10.1109/TSMCB.2011.2175726.

[4] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri and H. G. Jung, "A new approach to urban pedestrian detection for automatic braking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, IEEE, Dec, 2009, pp. 594–605. doi: 10.1109/TITS.2009.2032770.

[5] M. Jeong, B. C. Ko and J. Nam, "Early detection of sudden pedestrian crossing for safe driving during summer nights," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, IEEE, June, 2017, pp. 1368–1380. doi: 10.1109/TCSVT.2016.2539684.

[6] C. Godard, O. M. Aodha, M. Firman and G. J. Brostow, "Digging into self-supervised monocular depth estimation," *Proceeding of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, 2019, pp. 3828–3838.

[7] G. Jocher et al., "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Zenodo, Oct, 2020. doi:10.5281/zenodo.4154370.

[8] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, Sept, 2013, pp. 1231–37. doi:10.1177/0278364913491297

[9] T. Lin et al., "Microsoft COCO: Common Objects in Context," *Computer Vision – ECCV 2014*, Springer International Publishing, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

[10] M. Rezaei, M. Terauchi and R. Klette, "Robust vehicle detection and distance estimation under challenging lighting conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, IEEE, Oct, 2015, pp. 2723–2743. doi: 10.1109/TITS.2015.2421482.

[11] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, vol. 2, no. 6, San Francisco: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[12] NVIDIA TensorRT, Mar. 2021, [online] Available: https://developer.nvidia.com/tensorrt/.