Integrating Depth Cameras and Deep Learning for Enhanced Pedestrian Warning Systems

Daniel Lowe
Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
dlo54@uclive.ac.nz

***Abstract*—This paper proposes a method to track and predict pedestrians, including extended segments of pedestrians. This method applies the YOLOv8 deep learning algorithm to get the instance segmentation of pedestrians as masks. From these masks and a depth image, the masked depth of the pedestrian is retrieved. This masked image is then segmented by depth layer, and the contours of the depth layers are found. The segments and the whole pedestrian are tracked between frames, and a Kalman filter stores the real-world 2D depth and horizontal position of the pedestrian and segments. These Kalman filters are then used to predict the movement of the pedestrian and segments to see if they will overlap with a circle originating at the camera in the future.**

The proposed method could be run in real time at 19 frames per second, but the method was inaccurate as it had a false positive rate of at least 12%, and when it did correctly predict a future collision, its predictions were off by an average of 49% in time. Additionally, the method did not significantly improve upon a method where no depth segment tracking occurred, which achieved similar results.

These results indicate that the proposed method is not useful for pedestrian prediction over simpler methods.

I. Introduction

According to the traffic accident statistics compiled by the New Zealand Ministry of Transport, the number of accidents involving pedestrians in 2022 in New Zealand was 868. These accidents resulted in 34 fatalities, 240 serious injuries, and 595 minor injuries [1]. A study found that in 93.7% of accidents, the driver is at least partially at fault [2]. Therefore, systems that warn drivers about pedestrians could help prevent accidents.

In this vein, I propose a method to detect and predict pedestrian movement in real time by issuing a warning if a person will be within 3 metres of the vehicle using a vehicle-mounted Intel® RealSense™ D435 depth camera. This is achieved by depth-segmenting and tracking the segments of a person. However, I show no significant improvement in this method over simple whole-person prediction. Further details and the implications of these findings are discussed in the subsequent sections.

II. Background

There has been a lot of research into pedestrian avoidance, detection, and distance estimation. Because of this, many methods to detect and estimate the distance to a pedestrian have been proposed. These methods cannot deal with arbitrary extensions from the pedestrian in real time, but my proposed method can.

Yang et al. demonstrated a method to estimate the distance between a human and a robot by using a Kinect v2 camera [3]. They utilised the YOLO R convolutional neural network to identify humans and robot arm-bounding boxes, as well as a Kalman filter to increase confidence. From this, they obtained the centre x, y, and z coordinates of the bounding boxes and therefore could determine if the humans and the robot arm were too close. There are limitations to this method, though, as it cannot take into account the posture of the people or the industrial robot. It also assumes that the centre of the bounding box will contain the object being observed, which may not always be the case. For example, a person viewed from the side with their arms forward would not have pixels that are part of the person in the centre of their bounding box. In addition, this approach uses a stationary camera that is observing both subjects. This is not feasible for pedestrian detection for vehicles that may move over a large area. It also does not take into account the motion of people or the robot.

Yamaguchi et al. used a D455 depth camera and a laptop with YOLOv4 to generate bounding boxes from the colour channel of the frame for pedestrians and obtain the depth value at the centre of the bounding box from the depth channel. From this, they calculate the 3D coordinates of the centre of that bounding box and record the minimum depth inside that bounding box [4]. Each pedestrian is assigned an ID that is used to track them across frames with DeepSort. Then, the outliers are removed, and the next position of pedestrians is calculated based on their past trajectory, predicting potential collisions. This method still shares some of the same issues as Yang et al., particularly the reliance on the centreimitation is somewhat mitigated by the removal of outliers and the inclusion of the minimum depth in the algorithm, but the minimum depth value can be easily affected by noise or by objects occluding the person, so it may not be accurate. of a bounding box. This limitation is somewhat mitigated by the removal of outliers and the inclusion of the minimum depth in the algorithm, but the minimum depth value can be easily afprogrammeby noise or by objects occludng the person, so it may not be accurate. Its use of a camera on a bike is closer to the desired use case of a camera on a vehicle than Yang et al., and the addition of trajectory prediction allows the program

to foresee future collisions, instead of using a crude distance-based measurement.

Eguchi et al. [5] expanded upon Yamaguchi et al.'s approach by using an iPhone's LiDAR camera to obtain a depth image and its normal camera to capture a colour image, which was processed through YOLOv8 to generate bounding boxes for pedestrians. From this, the centre of the bounding box is detected. After correcting the depth image, the depth value for the centre of the bounding box is determined, and the coordinates of the pedestrian are established. Then it is calculated whether the pedestrian falls within a cone; if they do, it is deduced that they are in the danger range. This approach still has many of the same problems as those detected in the works of Yang et al. and Yamaguchi et al., due to its use of the centre of a bounding box. Although it uses a more practical camera and computing setup than the iPhone, its lack of trajectory prediction prevents it from forecasting future collisions.

Selmoune et al. [6] proposes a collision risk warning service procedure for both vehicle-vehicle and vehicle-pedestrian collisions. It works by using CCTV cameras to detect and classify objects as pedestrians, motorcycles, bicycles, vehicles, and personal mobility devices. From this, the objects are tracked using modified DeepSort where the Kalman filter is advanced, and instead of using the Hungarian algorithm, they used an original method. Then, the objects' overhead positions are found, and from this, trajectories are predicted with different models depending on the type of object. For pedestrians, their trajectories are predicted by classifying them as straight or curved depending on whether "the angle of the previous trajectories is smaller than the angle that we predetermine" [6], assuming that the pedestrian will continue on that path. These trajectories are calculated for an ellipse of possible start points centered on the pedestrian. If two trajectories are found to cross, a warning is sent. Unlike Yang et al. [3], Yamaguchi et al. [4], and Eguchi et al. [5], no depth camera is used. Instead, the method assumes vertical position does not matter and uses the fact that CCTV cameras are placed high up to get the position on the plane. This requires a specific position for the camera, but it allows the use of normal non-depth cameras. The more sophisticated pedestrian prediction enables a wider range of behaviors to be predicted than Eguchi et al. [5].

Qinghua et al. [7] proposed a method to predict human right hand movement for safe human-robot interaction. The method involved using the BP-HMT algorithm for human prediction by gathering training data of human hands in different scenarios performing different tasks and training from this. There are problems with this approach to human prediction; it requires labeled action types. In addition, compared to Eguchi et al., the prediction of human movement is a lot more computationally expensive than simple whole human trajectory prediction [5]. Moreover, if used as part of a warning system, a lot more of the human would need to be tracked and predicted, further increasing the required time. This method also focuses on short-term human prediction and may not be applicable to predicting human motion over the longer term, which is necessary for providing sufficient warning time.

Raffiei et al. [8] proposed a method to generate an optimal driving policy for pedestrian collision avoidance. This method involves using the DNQ algorithm, which approximates the optimal state-action using a neural network for Q-learning, in a simulated environment to train a function with a reward function defined by the vehicle maintaining a safe distance from pedestrians and the risk degree, which is defined by environmental conditions such as rain. The simulated vehicle can perform one of three actions: maintain current speed, slow down without brakes, or slow down with brakes. This method improves upon simple trajectory tracking used by Eguchi et al. [4], Yamaguchi et al. [5], and Selmoune et al. [6], with the authors finding that it improved upon existing methods in the simulation. However, its applicability to real-world situations is unknown, as there could be factors that the simulation did not take into account.

III. Proposed Method

A. Camera and Setup

An Intel® RealSense™ D435 camera was used to retrieve both a colour and depth image stream. It was operated at a resolution of 640 by 480 with a frame rate of 30 frames per second for both depth and colour streams. The D435 was selected for its reasonable accuracy for the desired ranges along with its affordability.

The computer used to process the video streams was running Linux Mint 21.1 Vera, and the data was retrieved from the camera and preprocessed so that the depth and the colour images matched using pyrealsense2. Python and OpenCV were used to process the data, and it was developed using PyCharm.

TABLE I: System Specifications

CPU	Intel® Core™ i7-9700G
GPU	GeForce RTX 2060
Camera	Intel® RealSense™ D435
Resolution	640 x 480 pixels
Frame Rate	30 frames per second
Operating System	Linux Mint 21.1 Vera
Python Version	3.9 [9]
OpenCV Version	4.9.0.80 [10]
NumPy Version	1.26.0 [11]
PyRealSense2	2.54.2.5684 [12]
FilterPy	1.4.5 [13]

B. Pedestrian Detection

The YOLO (You Only Look Once) architecture has been instrumental in pushing the boundaries of object detection performance, combining high precision with remarkable speed. The introduction of YOLOv8-nano or YOLOv8n, a more compact version of the YOLOv8 architecture, offers an optimal balance between detection accuracy and computational efficiency, making it particularly suitable for applications requiring real-time processing.

The proposed method uses a YOLOv8n convolutional neural network to detect humans in a 640x480px RGB image retrieved from the D435 camera. This model variant was

specifically chosen for its rapid inference capabilities and its proficiency in instance segmentation, the ability to distinguish between different instances of the same class. Utilising the pre-trained weights of YOLOv8n, which already recognises humans as one of its classes as it was trained on the COCO dataset [14], allows for a seamless incorporation into my pedestrian detection system without the necessity for extensive retraining but may not perform as well as a custom solution.

C. Depth Estimation Process

Following the detection of human instances by YOLOv8n, the method extends to analysing the depth of each detected pedestrian. This is achieved by employing the instance segmentation masks generated by YOLOv8n to isolate the areas corresponding to each detected human in the aligned depth image retrieved from the Intel® RealSense™ D435 camera. The median depth value within these masked regions is then calculated, providing a foundational depth estimate for further analysis. By using a median, I prevent outliers from affecting the measured distance of the person.

1) Depth Image Processing: To accommodate subsequent processing steps and compatibility with algorithms designed for standard 8-bit images, the depth image undergoes a scaling transformation. This transformation adjusts the depth values so that a distance of 7.5 metres corresponds to a pixel intensity of 255, effectively mapping the depth information to the 0-255 range typical of grayscale images, so that other algorithms can be applied to the depth image. This distance was chosen because 3 metres was the distance at which a person would be considered too close, so there is another 4.5 metres of distance where prediction and tracking can occur. As the fastest average walking speed in countries is 1.64 metres a second [15], this gives 2.7 seconds to predict movement, which is just under the 3 seconds used for later predictions. At the same time, this distance gives 2.9 centimetres per pixel, which is a long distance but not too long, so I believe this range is best for the method.

Further refinement of the depth data involves applying the instance segmentation mask to the scaled depth image, where pixels outside the masked regions are set to the maximum value of 255. This step ensures that only the relevant areas corresponding to detected pedestrians are considered in subsequent analyses.

2) Depth Segmentation for Extended Segment Detection: To identify and evaluate the presence of extended segments of a pedestrian, such as outstretched arms, which might not be immediately detected in the initial detection phase, the method utilises a depth segmentation technique. This involves segmenting the scaled-depth image into discrete layers based on depth ranges.

Each layer is preprocessed by performing three opening and closing morphological operations with a kernel size of three to remove small details, creating a mask like Figure 2. If these closing and opening morphological operations are not performed, then numerous small sections will be found, as in Figure 1, which slows the method and prevents it from running in real-time. Additionally, it reduces noise and prevents small outlying regions from affecting the measurement.

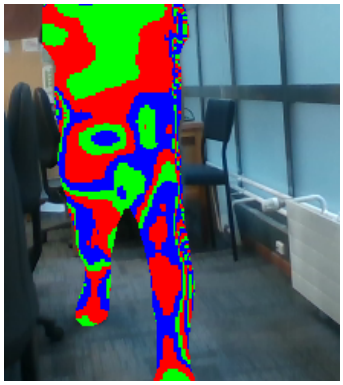

Fig. 1: Human's depth segments without morphological operations

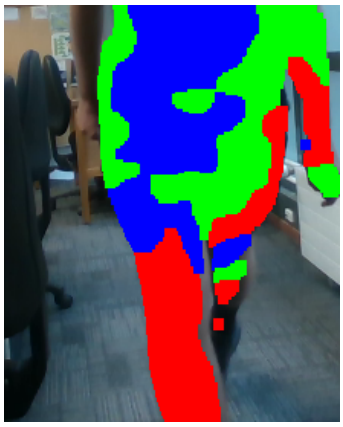

Fig. 2: Human's depth segments displayed

I chose three opening and closing morphological operations specifically because operations above that did not increase the performance to a significant degree and caused it to sacrifice detail in what it was tracking, and the performance for 3 operations of 51.4 milliseconds allows 19 frames per second to be processed, which is sufficient for applications requiring real-time processing.

This is followed by using Suzuki et al.'s border following algorithm [16] to get the contours, which is both effective and efficient. For example, Figure 3 shows contours obtained from applying this to a pedestrian's leg.

3) Final Depth Estimation: For each detected contour, the median depth value is calculated from the non-scaled depth data and compared to the overall median depth of the pedestrian. If the disparity in depth is less than 0.92 metres (chosen as it is half of the average height in the Netherlands, the world's tallest country, as of 1980 [17]), then the contour is tracked. The minimum of these tracked median depth values is then used as the depth value if it is not more than the median depth value for the person. The result of this can be seen in Figure 4, illustrating the contour of the pedestrian segment nearest to the camera.

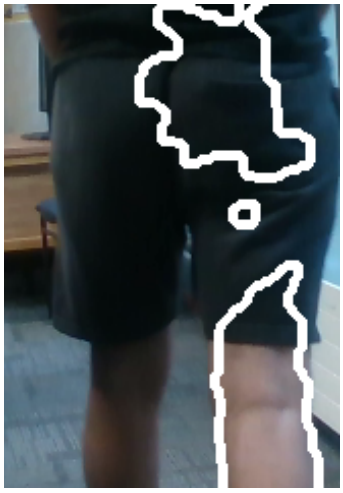

Fig. 3: Contours of Depth Segmentation for a human leg

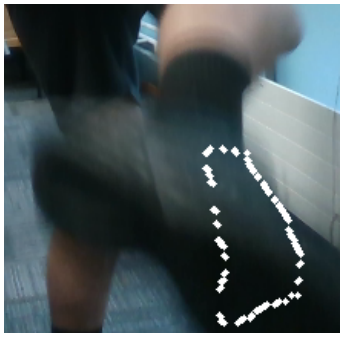

Fig. 4: Closest contour of a foot

4) Position Estimation: Using this depth and the centre of the bounding box of either the contour or mask from which that distance estimation is taken, the world coordinates are calculated. It is assumed that the camera is level with the ground, so if the xz coordinates are in a circle that is defined as on the xz plane centred at the camera with a radius equal to the safe distance, then the human is in the warning area and a warning notification is issued.

D. Trajectory Prediction

1) Pedestrian Tracking: The process of pedestrian tracking begins with the identification of pedestrians in the image. Their bounding boxes and median distances are then compared with those of pedestrians that appeared in the previous frame. This facilitates the calculation of the distance between each newly detected pedestrian and their corresponding pedestrian from the last frame. Subsequently, each pedestrian is assigned to the pedestrian with the minimum distance from them. This approach leverages both depth information and bounding box data to distinguish individuals, with the aim of identifying pedestrians in scenarios where pedestrians' bounding boxes overlap.

2) Segment Tracking: In conjunction with pedestrian tracking, the method extends to segment tracking. Here, each segment detected during the "Depth Segmentation for Extended Segment Detection" phase is associated with a bounding box. Leveraging the segments detected within the same pedestrian in the previous frame, the bounding box and depth algorithm are again employed to ascertain the segment that best aligns with the current segment under consideration, since segments may fragment; one segment old can be associated with multiple current segments.

3) Prediction: Both pedestrians and segments that are being tracked have Kalman filters that are stored throughout the frames or copied to allow fragmentation in the case of segments, and that are updated with new measurements every frame. Kalman filters are a method of interpreting and predicting data with noise, as proposed by Kalman et al. [18].

The Kalman filters contain a 2D representation of the object's position and velocity, with the dimensions being the deprojected distance from the camera and the projected horizontal direction from the camera.

The 2D representation was chosen because vertical distance does not matter, both in that pedestrians can't fly, so their movement is limited to the ground, and in that vertical distance does not matter if something is within an unsafe range. There is an assumption made here that the camera is level with the ground,, as it assumes that the horizontal direction is not the vertical direction.

If a Kalman filter has not seen at least 5 frames, it is ignored because otherwise the noise from the initial data can cause too many false positives. Otherwise, its position in three seconds is calculated from the predicted position and velocity; this is used along with its current position to create a line segment. Then this line segment is compared to a circle with a radius of danger distance metres centred at the camera, and if it overlaps, then the method would issue a warning.

IV. Results

A. Evaluation Method

To evaluate the effectiveness of my method, I placed the Intel® RealSense™ D435 camera at point D facing towards d on a 10 by 20-metre area as described in Figure 6. From here, I recorded myself walking from all points a, b, c, d, and e to all points A, B, C, D, and E, giving a total of 25 separate recordings. These recordings were done on a sunny day in a grassy field with trees and buildings in the background, as shown in Figure 5.

Fig. 5: Location of experiment

These recordings were then processed through the method on a computer with specifications given in Table I, and the

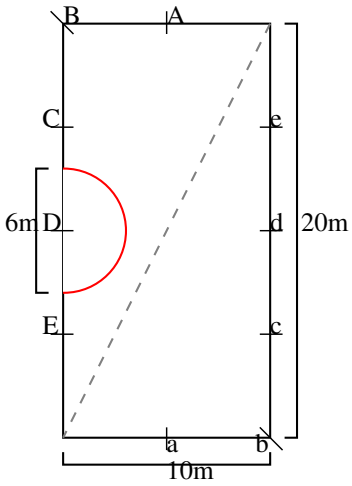

Fig. 6: Experiment layout

times when it predicted that I would be within a distance of three metres of it in the next three seconds were recorded and compared to manually observed values. For the predictions, I used what the recording showed in the future to see if they were valid or not, if they were delayed or ahead of reality, and by how much, by comparing them to the recordings run through the method and a modified version of the method with segment prediction removed.

B. Reliability

None of the recordings resulted in a false negative outcome, but four cases where no intersection was observed occurred: bC, bE, dE, and eE. The same was true for the one without segment prediction, other than the fact that only bE, dE, and eE were found to be false positives. bC theoretically should intersect the circle, so it can't be called a false positive. It is likely the case that the camera did not catch the instance when they intersected the circle, but bE, dE, and eE all should not theoretically intersect the circle. Moreover, if you assume all paths that should intersect the circle did, then aB and aC should predict a collision. These could be interpreted as false negatives, but it is also probable that my path was not perfectly straight, so these were not recorded as that. This is also the case when no segment prediction is applied. Consequently, this results in the false positive and false negative counts shown in Table II.

TABLE II: False positive and false negative counts

	Range Upper	Range Lower
False positive (with segment tracking)	4	3
False negative (with segment tracking)	2	0
False positive (no segment tracking)	3	3
False negative (no segment tracking)	3	0

Only four instances of a true positive prediction could be recorded, which are shown in Table III. eD and aD are not in this table because they stayed outside the camera's field of view, so they could not be predicted. This gives a mean absolute value of 1.46 seconds and 1.47 seconds without the extended segment prediction, for the amount that predictions are off the observed times. This is large, approximately half of the predicted time delay for both of them.

TABLE III: Time Discrepancy: Actual vs. Predicted Overlap (Seconds)

Path	With Segment Tracking,	No Segment Tracking
aC	-0.488	-0.690
bD	2.24	2.35
cD	1.53	1.54
dD	1.57	1.30

When compared to Yamaguchi et al., which used a different method to test its method and defined an appropriate warning as being 2.1 seconds plus or minus 1 second, they found that 100% of warnings fall within this range and no false positive or false negative were thrown [4]. If I were to define a similar rage of 3 plus or minus 1.5 seconds, then I would have 3 out of 4 not giving reasonable predictions with segment tracking and 2 out of 4 with no segment tracking. This shows that my method is less accurate than Yamaguchi et al.'s method. As well as that my method has false negatives which Yamaguchi et al.'s method did not have.

C. Efficacy

The values for the time taken per frame was retrieved by running my method though the collected video with out displaying anything and when it detected a pedestrian I stored the time taken to process the frame, and then averaged it. All measured average times are shown in the Table IV and the difference between all pairs of frame rates are statically significant with a p value of at most 0.05.

TABLE IV: Effects of opening and closing operations on performance

Opening and Closing Count	Mean Time Taken per Frame (Milliseconds)
0	558
1	119
2	64.4
3	51.4
4	45.2
5	43.0

D. Limitations

There is an implicit assumption that the camera is level to the ground, as if this is not the case, its warning range circle will not trace a circle on the ground but an ellipse, and that ellipse will have some offset compared to the true centre of the camera. This can all result in miscalculations on whether something presently exists or will be too close. If the ground it was placed on was not level or I tilted the camera, then my results may not be accurate.

In addition, based on the limited data obtained from the experiment conducted on the method's effectiveness. the fact that the data I do have is not representative of how real

pedestrians behave, drawing definitive conclusions about the effectiveness of the method is not possible.

Although I have another data to make statistically significant claims about performance the problem of my collected footage not being real persists so I don't know if it will have a similar performance in real use.

In addition, all experiments are run on a sunny day in a grassy field so the method may behave differently in different environments and weather conditions.

Finally, my comparison to Yamaguchi et al. may not be completely valid as the methods used are not the same. So Yamaguchi et al.'s method may not have the same results as it did in my method if it were tested using my method.

V. Conclusion

In this paper, I propose a method to detect, track, and predict pedestrians so that a warning could be sent if they approached a certain radius of the camera.

This method used the YOLO8n convolutional neural network to do instance segmentation of pedestrians, which was then used with the depth image from the depth camera to track segments of the person that appear extended in the depth camera by depth segmentation as well as the whole person. Using Kalman filters, these segments and whole pedestrians had their movement predicted, and if they would cross the circle around the camera, then the method would return a warning.

This method was inaccurate as it exhibited a false positive frequency of at least 12%, and when it did correctly predict a future conclusion, its predictions were off by an average of 49% in time. While it could be executed in real-time at 19 frames per second, this was only on a powerful computer, so it may not be real-time on a less powerful computer. In addition, the segment tracking did not demonstrate a notable enhancement over the simpler method of not including it. In addition, when it was compared to Yamaguchi et al.'s method, it yielded less accurate predictions then it, with only 1 out of 4 being within the reasonable range, as defined by Yamaguchi et al. compaired to 5 out of 5 [4]. As well as that Yamaguchi et al.'s method did not have false positives, unlike my own.

To the best of my knowledge, this precise method of pedestrian prediction where depth segmentation is used to predict parts of a person has not been published before, but many other methods of pedestrian prediction have been utilised and suggested in academic literature.

A. Future Research

Additional testing could be conducted to evaluate the method's performance with real pedestrians. This could be achieved through the acquisition of footage featuring real pedestrians captured by depth cameras, followed by an evaluation of the algorithm's performance. This would allow the method to be more accurately assessed in a more complex environment.

The prediction of the extended segments could be made more realistic by modelling them with a more complex model. For example, a constraint could be imposed on the extended segments where they cannot extend over a certain distance from the predicted position of the overall pedestrian. This measure could reduce false positives while maintaining the generality of the algorithm that allows all segments of the human body to be tracked. The model could also be extended so that it models each part differently. For example, legs are more accurately represented as rotating objects with springs on the end [19].

The segmentation algorithm could undergo training to include held items as components of a pedestrian's segmentation mask. This would allow the algorithm to issue a warning if either the person or what they are holding would be within the range of the camera, which may be useful at worksites where people may be carrying long boxes or similar items.

A gyroscope or alternative method could be used to determine the downward direction, meaning that the requirement for the camera to be level with the ground could be removed.

To allow the warning to be sent regardless of the angle from which a pedestrian is approaching, a 360-degree camera could be used.

References

[1] T. M. W. . T. M. of Transport, "Te Marutau — Ngā tatauranga ā-tau — Safety — Annual statistics — transport.govt.nz," https://www.transport.govt.nz/statistics-and-insights/safety-annual-statistics/sheet/pedestrians, 2023, [Accessed 14-03-2024].

[2] G. Zhang, K. K. Yau, and X. Zhang, "Analyzing fault and severity in pedestrian–motor vehicle accidents in china," *Accident Analysis Prevention*, vol. 73, pp. 141–150, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457514002553

[3] H. Yang, Y. Tang, and W. Yu, "A human-machine safety distance detection method based on computer vision," in *2022 34th Chinese Control and Decision Conference (CCDC)*, 2022, pp. 5784–5789.

[4] N. Yamaguchi, H. Tokumaru, O. Fukuda, H. Okumura, and W. L. Yeoh, "Bicycle-based collision prevention system using pedestrian trajectory prediction," in *2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW)*, 2022, pp. 151–153.

[5] M. Eguchi, N. Yamaguchi, O. Fukuda, H. Okumura, and W. L. Yeoh, "Bicycle-based collision prevention system using iphone with lidar," in *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*, 2023, pp. 358–360.

[6] A. Selmoune, J. Yun, M. Seo, H. Kwon, C. Lee, and J. Lee, "Development of a residential road collision warning service based on risk assessment," *Journal of Advanced Transportation*, vol. 2023, p. 7496377, Mar 2023. [Online]. Available: https://doi.org/10.1155/2023/7496377

[7] Q. Li, Z. Zhang, Y. You, Y. Mu, and C. Feng, "Data driven models for human motion prediction in human-robot collaboration," *IEEE Access*, vol. 8, pp. 227 690–227 702, 2020.

[8] A. Rafiei, A. O. Fasakhodi, and F. Hajati, "Pedestrian collision avoidance using deep reinforcement learning," *International Journal of Automotive Technology*, vol. 23, no. 3, pp. 613–622, Jun 2022. [Online]. Available: https://doi.org/10.1007/s12239-022-0056-4

[9] (2023) python.org. [Online]. Available: https://www.python.org/downloads/release/python-3918/

[10] (2023) OpenCV: OpenCV modules — docs.opencv.org. [Online]. Available: https://docs.opencv.org/4.9.0/

[11] (2023) NumPy - News — numpy.org. [Online]. Available: https://numpy.org/news/

[12] (2023) Release Notes — github.com. [Online]. Available: https://github.com/IntelRealSense/librealsense/wiki/Release-Notes

[13] (2018) filterpy — pypi.org. 16-04-2024. [Online]. Available: https://pypi.org/project/filterpy/

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[15] R. V. Levine and A. Norenzayan, "The pace of life in 31 countries," *Journal of Cross-Cultural Psychology*, vol. 30, no. 2, pp. 178–205, 1999. [Online]. Available: https://doi.org/10.1177/0022022199030002003

[16] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X85900167

[17] M. Roser, C. Appel, and H. Ritchie, "Human height," *Our world in data*, 2013.

[18] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[19] R. M. Alexander, "Simple Models of Human Movement," *Applied Mechanics Reviews*, vol. 48, no. 8, pp. 461–470, 08 1995. [Online]. Available: https://doi.org/10.1115/1.3005107