Lightweight Collaboration of Detecting and Tracking Algorithm in Low-Power Embedded Systems for Forward Collision Warning

Sunghoon Hong[1,2] and Daejin Park[2*]
[1]CARNAVICOM, Incheon, Republic of Korea
[2]School of Electronics Engineering, Kyungpook National University, Daegu, Republic of Korea
[*]Correspondence to: Daejin Park (boltanut@knu.ac.kr)

***Abstract*—The cause of the majority of vehicle accidents is a safety issue due to the driver's inattention, such as drowsy driving. A forward collision warning system (FCWS) can significantly reduce the number and severity of accidents by detecting the risk of collision with vehicles in front and providing an advanced warning signal to the driver. This paper describes a low power embedded system based FCWS for highway safety. The algorithm described in this paper computes time to collision (TTC) through detection, tracking, distance calculation for the vehicle ahead and current vehicle speed information with a single camera. Additionally, in order to operate in real time even in a low-performance embedded system, an optimization technique in the program with high and low levels will be introduced. The system has been tested through the driving video of the vehicle in the embedded system. As a result of using the optimization technique, the execution time was about 170 times faster than that when using the previous non-optimized process.**

***Index Terms*—Forward collision warning system, object detection, low-power vision processing, hardware-software acceleration**

I. Introduction

An FCWS detects the risk of collision with the vehicle in front and sends an alert signal to the driver in the form of an audio alert, visual pop-up display, or other warning alert. An FCWS can contribute significantly to reducing the number and severity of driving accidents by sending an alert to the driver about a possible impending collision if the vehicle gets too close to the vehicle in front of it. TTC is computed based on the speed of the current vehicle and distance from the vehicle in front of it. To calculate the distance to the vehicle in front, detection and tracking algorithms for the vehicle in front are required. In vehicle detection, a powerful detection algorithm such as deep learning cannot be applied in a low-performance embedded system.

Therefore, in this paper, we use a machine learning technique for faster detection. The Haar feature-based cascade classifier [1], which is a kind of machine learning, provides effective object detection. However, these algorithms are also limited in application to low-performance embedded systems, so they need to be optimized. A vehicle tracking algorithm is required to calculate the distance for the detected vehicle, and a template-matching technique [2] was used for vehicle tracking. To quickly detect a vehicle intervening in front, the vehicle detection algorithm must always operate in real time. However, if vehicle tracking and detection are performed at the same time, there is a concern that the performance of the FCWS will degrade because the computational complexity increases.

In this paper, section II discusses the related works in FCWS. Section III introduces a method of optimizing this algorithm at the high and low levels, and shows that the performance of FCWS is much improved compared with the previous one in section IV. Finally the conclusion is mentioned in section V.

This work was supported by the Technology Innovation Program (P0013847, 20%, Development of automatic steering-based accident avoidance system for electric-driven port yard tractors operating at low speed (less than 30 km/h)) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2019R1A2C2005099, 10%), Ministry of Education (NRF-2018R1A6A1A03025109, 20%) and partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00944, Metamorphic approach of unstructured validation/verification for analyzing binary code, 50%)

II. Related works

From a technological perspective on FCWSs, a fusion of Radar and vision or Lidar and vision is an attractive approach. In such a system, the Radar or Lidar gives accurate range and range rate, while the vision detects the vehicle. However, this solution is expensive and it is not easy to match the sensors well. On the other hand, a single vision system has the advantage in terms of cost, because it does not require a separate sensor-matching operation, and is simple to install. There are many ways to detect vehicles in a single vision system. The method of detecting a vehicle using only edge information [3] results in many false detections. To overcome this, there is a deep learning technique by which vehicles are detected using YOLOv3 [4].

Deep learning provides good detection performance, but the disadvantage is that it cannot be applied to a low-performance embedded system because a large amount of computation is required. Machine learning techniques such as vehicle detection using cascade classifier [5] are faster than deep learning techniques are. As mentioned before, optimization techniques are important to increase the operating performance in low-performance embedded systems. There are various optimization techniques. The high-level techniques have been introduced such as loop optimization using loop unrolling and response speed improvement using thread. However, because there is a limit to optimization at a high-level, an optimization

technique at a low-level using single instruction, multiple data (SIMD) or multi-core processing is required.

III. Proposed Architecture

The final goal of FCWS is to send an alert signal to the driver about a vehicle collision by calculating the TTC. Vehicle detection, tracking, and distance calculation algorithms are required to calculate TTC. Because the vehicle detection and tracking algorithms require a lot of computational processing, the embedded system needs to be optimized as shown in Fig.1.

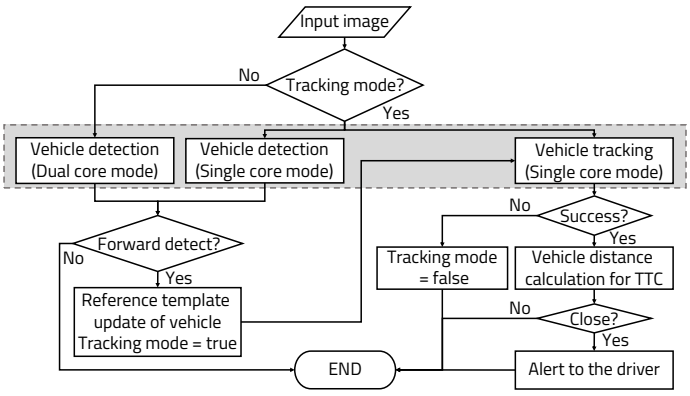

Fig. 1. FCWS flowchart

However, because there is a limit to optimization at the high-level, a technique for optimizing at a low-level is required. SIMD helps to improve operation speed by performing the same operation on multiple data operands concurrently.

A. Vehicle Detection

1) Haar feature-based cascade classifiers: The vehicle detection algorithm uses Haar feature-based cascade classifiers. The Haar feature-based cascade classifiers are based on machine learning and have the advantage of being lighter and faster than deep learning. Haar features are a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle just like convolutional kernels. To extract only meaningful features in vehicle detection, Adaboost is used. Finally, the cascade function is trained from numerous positive images (images of vehicles) and negative images (images without vehicles) in a 20x20 window size. It is then used to detect vehicles in other images as shown in Fig.2.

Instead of applying all meaningful features to a 20x20 window, the features are grouped into different stages of classifiers and applied one-by-one. If a window fails in the first stage, it is discarded. If it passes, the second stage of features is applied and the process continues. The window that passes all stages is a vehicle region. However, the problem in the cascade function is to repeat sliding and increasing all windows in the input image. In fact, the window, such as the background is completely different from the vehicle, so even calculating it in the first stage of the cascade is inefficient. The vanishing point helps to use only meaningful windows for detecting vehicles in input images.

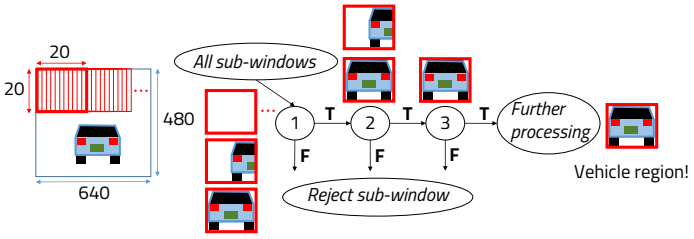

Fig. 2. Cascade windows sliding system

2) Vanishing point based cascade classifiers: A vanishing point is a point on the image plane where the two-dimensional perspective projections of mutually parallel lines in three-dimensional space meet at one point. In the case of a detected vehicle, because the coordinates of the part touching the ground cannot exist above the vanishing point, it is calculated by dividing the windows of 6 steps as shown in Fig.3.

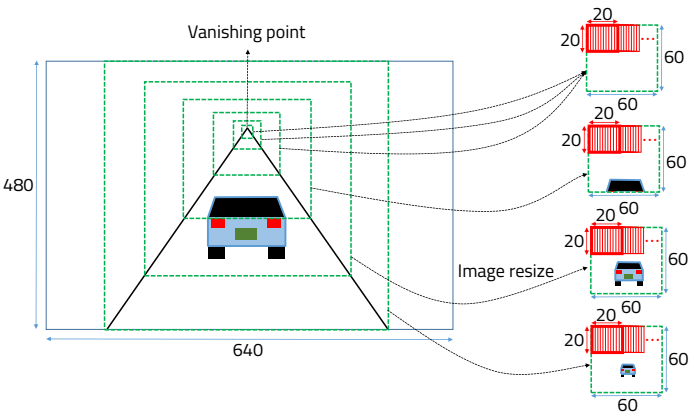

Fig. 3. Vanishing point based cascade system

If all windows of 6 steps are processed simultaneously in one frame (image), the processing speed is slow and the desired frame rate cannot be guaranteed. At a high-level, to improve this part, one step per frame is calculated, that is, 6 frames are required to calculate all windows of 6 steps.

3) SIMD: SIMD processing elements that perform the same operation on multiple data points simultaneously. Using this method, vehicle detection speed is improved.

B. Vehicle Tracking

A vehicle tracking algorithm is necessary to continuously track a previously detected vehicle. In the case of only detecting without tracking, the pixel coordinates of region of interest (ROI) for the detected vehicle are irregularly different. If the coordinates are irregularly different, the distance value also changes irregularly, so the TTC has a large error. To constantly track the detected vehicle in front, a tracking algorithm is required, and the vehicle tracking algorithm uses a template matching technique in 20x20 window.

However, if only the detected vehicle in front is tracked, recognition may be slow when a new vehicle intervenes. In addition, if vehicle tracking and detection are computed

simultaneously, the computational complexity increases, and vehicle tracking may fail in real time. To solve this problem, we use a multi-core processor, one core computes to detect the vehicle and the other core computes to track the vehicle. When detecting a vehicle by using one core, the vehicle is detected in 6 steps of the window using the existing method.

C. Distance Computation

The process of calculating the distance from the vehicle ahead is the most important step to calculate the TTC. The distance value is computed based on a pinhole camera model as shown in Fig.4.

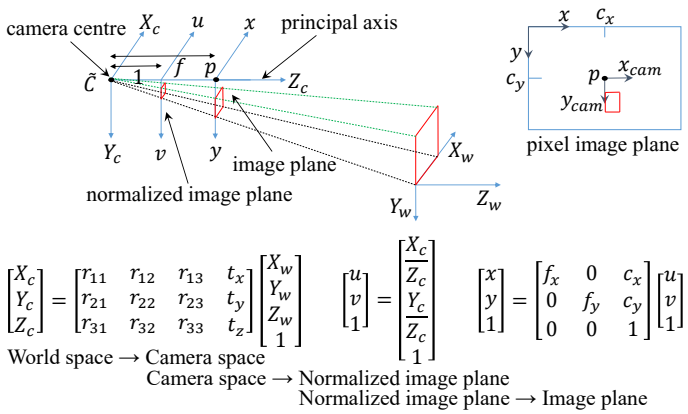

Fig. 4. Pinhole camera model

The pinhole camera model describes the mathematical relationship of the projection of points in three-dimensional space onto the image plane of an ideal pinhole camera. The distance value is computed in the tracking step after a vehicle is detected as shown in Fig.5.

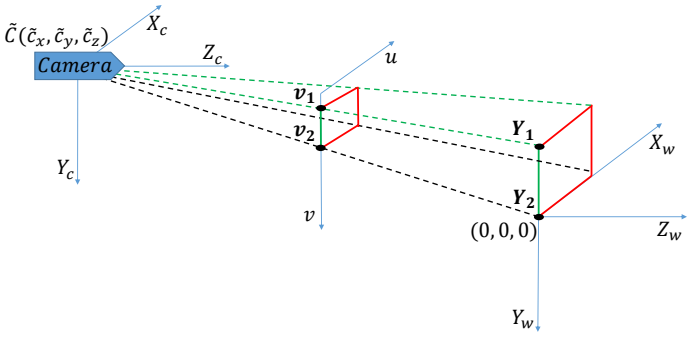

Fig. 5. Forward vehicle distance calculation

Assuming that the camera has only the rotational component of θ, a point in the world space will project to the normalized image plane at a point, where a point in the normalized image plane is given by the Equation 1.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = s \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & \cos\theta & -\sin\theta & t_y \\ 0 & \sin\theta & \cos\theta & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where θ is camera tilt angle and s is scale factor to project a point in the world space onto the normalized image plane. The rotation matrix (R) and the translation vector (T) are transformation matrix that move a point from the world space to the camera space as shown in Equation 2.

$$P_c = R(P_w - \tilde{C}) = RP_w - R\tilde{C} = RP_w + T \tag{2}$$

where $\tilde{C}$ is a central point of the camera in the world space. Translation vector is computed by the Equation 3, 4.

$$T = -R\tilde{C} \tag{3}$$

$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = - \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \tilde{c}_x \\ \tilde{c}_y \\ \tilde{c}_z \end{bmatrix} \tag{4}$$

A point in the normalized image plan will be defined as shown in Equation 5.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = s \begin{bmatrix} X - \tilde{c}_x \\ Y\cos\theta - Z\sin\theta - (\tilde{c}_y\cos\theta - \tilde{c}_z\sin\theta) \\ Y\sin\theta - (\tilde{c}_y\sin\theta + \tilde{c}_z\cos\theta) \end{bmatrix} \tag{5}$$

When the Y_2 is an origin point in the world space we will define the v_2 as shown in Equation 6.

$$v_2 = \frac{-\tilde{c}_y\cos\theta + \tilde{c}_z\sin\theta}{-\tilde{c}_y\sin\theta - \tilde{c}_z\cos\theta} \tag{6}$$

where $\tilde{c}_y$ is the camera height mounted on vehicle and $\tilde{c}_z$ is the distance from the camera to detected vehicle. It is than possible to compute the distance from camera to detected vehicle as shown in Equation 7.

$$\tilde{c}_z = \frac{f_y\cos\theta - (y_2 - c_y)\sin\theta}{(y_2 - c_y)\cos\theta + f_y\sin\theta}\tilde{c}_y \tag{7}$$

where f_y is the camera focal length of y-axis and c_y is the camera principal point of y-axis. Y_2 is a point of contact between the vehicle and the road in three-dimensional space, and the projected point on the image plane is y_2. The distance $\tilde{c}_z$ from the vehicle in front can be calculated from a point where the vehicle and the road contact each other in the image plane.

IV. Implementation and Experimental Result

In the test, the speed measurement of the algorithm was compared through the driving video previously stored in the LS1028A [6] board. The LS1028A is a processor made by the NXP company and is equipped with two powerful 64-bit Armv8 processors. The ARM NEON technology provided in the Armv8 processor allows optimizations using SIMD's 128-bit operation registers. Another feature is that the GPU is built-in, but since most low-power embedded systems do not have a built-in GPU, the GPU is not used in this paper. The method used in the non-optimized cascade algorithm takes about 5.192 seconds to detect a vehicle, but if optimized through SIMD, it can be seen that the detection is faster to

about 4.416 seconds as shown in Fig. 6. The method used in the optimized vanishing point based cascade algorithm takes about 0.034 seconds and if optimized through SIMD, it can be seen that the detection is faster to about 0.030 seconds as shown in Fig. 7.

The total frame execution time of the 3356 frames according to vanishing point based cascade using SIMD is about 170 times faster than non-optimized cascade algorithm as shown in Fig. 8. Vehicle tracking can also been further improved as a result of using SIMD as shown in Fig. 9. If there is no vehicle in front or a new reference template is updated, it can be seen that the execution time is zero because vehicle tracking algorithm is not performed, only the vehicle detection algorithm is performed. In this way, incorporating FCWS into a vehicle is effective when using SIMD because low power and processing speed are important.

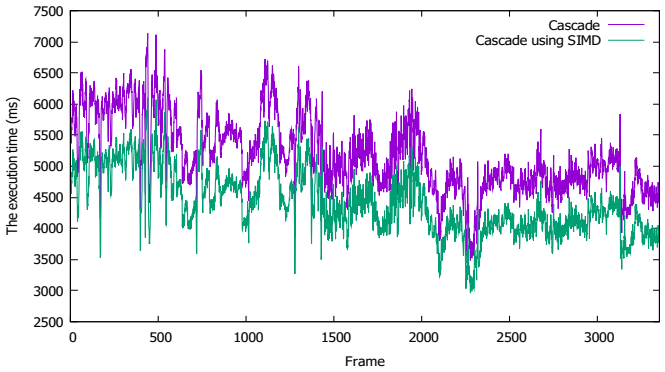

Fig. 6. The execution time according to the number of frame in the non-optimized or optimized cascade algorithm

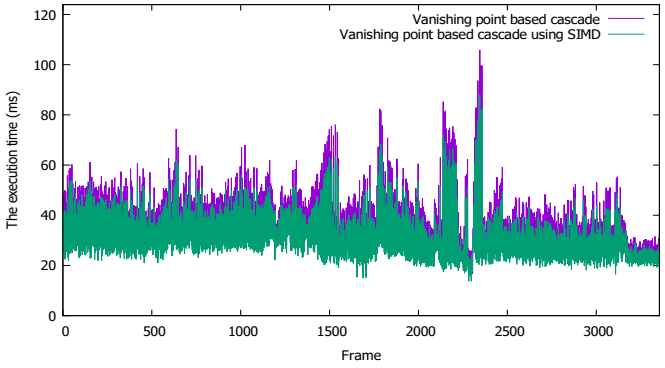

Fig. 7. The execution time according to the number of frame in the non-optimized or optimized vanishing point based cascade algorithm

V. Conclusion

We have presented an FCWS based on fidelity controllable optimization techniques using the consideration of trade-off in terms of computation complexity and detection accuracy. Image information coming from the single camera is used to detect and track the vehicle. Existing methods had limitations

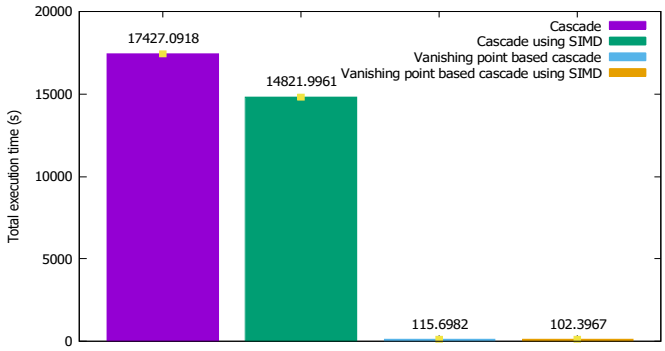

Fig. 8. The total execution time according to vehicle detection algorithms

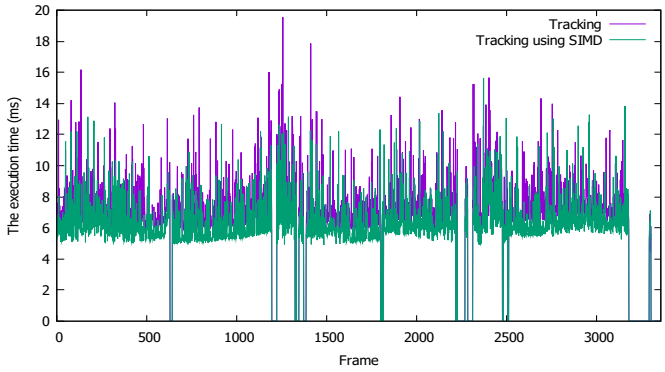

Fig. 9. The execution time according to the number of frame in the non-optimized or optimized tracking algorithm

because they optimized only at the high-level without considering the low-level. The system has been tested through the driving video of the vehicle, and it can be seen that the processing speed of vehicle detection and tracking has been improved compared with the previous method through optimization. Through optimization, the power consumed by the embedded system is also reduced, which helps with vehicle fuel economy. As such, a low power based embedded system is important.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I.

[2] S. Omachi and M. Omachi, "Fast template matching with polynomials," vol. 16, no. 8, 2007, pp. 2139–2149.

[3] S. A. Nur, M. M. Ibrahim, N. M. Ali, and F. I. Y. Nur, "Vehicle detection based on underneath vehicle shadow using edge features," in *2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, 2016, pp. 407–412.

[4] S. Zhang, L. Chai, and L. Jin, "Vehicle detection in uav aerial images based on improved yolov3," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2020, pp. 1–6.

[5] M. A. Zulkhairi, Y. M. Mustafah, Z. Z. Abidin, H. F. M. Zaki, and H. A. Rahman, "Car detection using cascade classifier on embedded platform," in *2019 7th International Conference on Mechatronics Engineering (ICOM)*, 2019, pp. 1–3.

[6] NXP, "https://www.nxp.com/docs/en/fact-sheet/ls1028afs.pdf," in *Layerscape LS1028A Family of Industrial Applications Processors*.