Bus Crowdedness Sensing Based on Deep Learning

Wenhao Huang, Akira Tsuge, Yin Chen, Tadashi Okoshi and Jin Nakazawa
Keio University
5322 Endo, Fujisawa, Kanagawa 252-0882, Japan

***Abstract*—This research proposes a sensing system to collect bus crowdedness data by detecting passengers' getting off and on events in a bus via deep learning-driven image processing on video data of passengers. The system is designed to be deployed to and conduct real-time object recognition and object tracking on a bus. In our prototype system, object recognition is implemented based on yolov3 with Darknet53, and object tracking is implemented by combining object recognition, Kalman Filter (LQE) and Hungarian Algorithm. The performance of the system is evaluated experimentally using driving recorder video data taken from a bus.**

***Index Terms*—Object Detection, Bus Sensing, Deep Learning, Smart Cities, Image Processing.**

I. Introduction

In 2020, COVID-19 pandemic has greatly affected people's lives over the whole world. Social distancing is widely considered as an important practice to slowing down the spread of the COVID-19 virus. While we can avoid human contact in many situations, it is hard to keep sufficient distance in crowded public transportation vehicles. Most public transport operators have put forward some common countermeasures [1], but there is still no effective way to avoid the problem of crowded public vehicles. Congestion means crowded buses, the crowding means closer social distance, which makes it more susceptible to infection virus. For a bus service, besides the major goal of carrying passengers around, providing a comfortable and safe travel experience for passengers is also an important business consideration. Bus comfortability contains many aspects, among which passenger density inside a bus is the most important one. The crowdedness can directly affect the comfortability of people in the bus [2]. There is increasing interest in crowdedness to reduce the risk of infection in the COVID-19 environment. In a survey of the use of high-speed buses, "In order to prevent coronavirus infection," "because of the risk of infection in cramped vehicles," and other concerns about taking the bus gathered. Many people feel uneasy about being unable to avoid a "dense" environment for a long time. **Figure 1** summarizes the result of the survey of taking a bus and it is notable that there is a lot of concerns about bus environment [3].

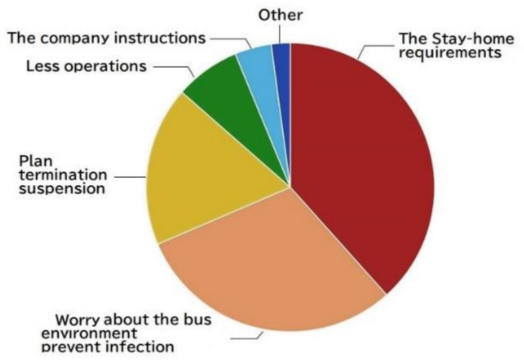


Fig. 1. In the survey, the question "Why did you use the highway bus less often?" was asked. In the answers to the survey, there were many concerns about getting on the bus, such as "to prevent coronavirus infection" and "because there is a risk of infection in a small bus". It seems that many people are worried about the environment where "denseness" cannot be avoided.

From the current practical situation, it is a feasible method to obtain the crowdedness degree directly by measuring the number of people through the camera inside the bus and control the crowdedness with the data of passenger's density. Traditional approaches to obtain passenger density rely on field investigation. There are two drawbacks. First, it involves expensive labor efforts. Nevertheless, field investigation is not scalable: it is difficult to maintain a record staff all day in every station. Second, it is hard to identify the whole trip of a single passenger. As a consequence, it is hard to estimate the effectiveness of the new schedule [4]. With the development of technology, machine learning, and computer vision came out, methods of counting people by image processing have been continuously proposed [5,6,7,8]. In [5,6,7] the images were analyzed and processed directly to obtain the results. In [8], the machine learning method was used to calculate the number of people through the trained model.

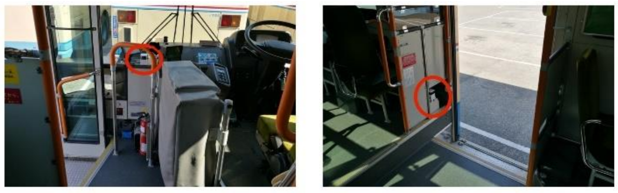
In the above research, it can be seen that relevant technologies have been relatively mature, but many methods have limitations on the camera angle and the special features acquired. We propose a method that can detect in real time the crowdedness of a bus and can be applied to all images and environments in a bus. On the one hand, it can make passengers feel more safely to choose the right time to get on the bus and know the degree of crowdedness. On the other hand, it can also tell the operating company to allocate the vehicle resources reasonably. In this study, we will discuss, experiment, and compare the realization methods of the crowdedness detection.

The rest of paper is organized as follows. Section II will make a briefly review of related work. Section III will introduce the image processing algorithm. Section IV will show the experimental evaluation and conclude this paper in Section V.

II. Related Work

A. *Crowdedness Detection*

About the crowdedness detection, there has been much research realized it. In the implementation of the methods are different, mainly divided into image analysis [9] and sensors [10] to achieve. Tracklet and Mean-shift are used in [9] to track the special spot of people, so as to realize the number detection. It also has a good performance in the results. In the study of [10], the detection and counting were carried out through the LiDAR sensor. The position of the sensor was set like **Figure 2**, and the information obtained from the environment is only three coordinates within a fixed range, that was relatively single. We will also use image processing in this project, but it is a little different from [9]. We do this via object detection.


(a) Device of front door (b) Device of rear door
Fig.2. Installation of measuring device [10]

B. *Object Detection Model*

Object recognition using Darknet network or Faster RCNN network combined with YOLO [11] is the most commonly used technique in point-to-point real-time target detection. The YOLO model is quite easy to use, has great performance, and has a good ecology. It is called an end-to-end object detection method in the sense that the deep neural network handles it from the beginning to the end. By making it end-to-end, you can benefit from optimization by deep neural network even when extracting a region with high objectness, and because the bottleneck of multi-stage configuration is eliminated, it is faster and higher. Nowadays, an accurate and fast detection method has become possible. It's also easy to see the results in the **Figure 3**. This is also why it becomes a popular implementation method.

We will use the above-mentioned YOLO model as the basis of our target detection in this study. We will explain how we implement this in the next section.

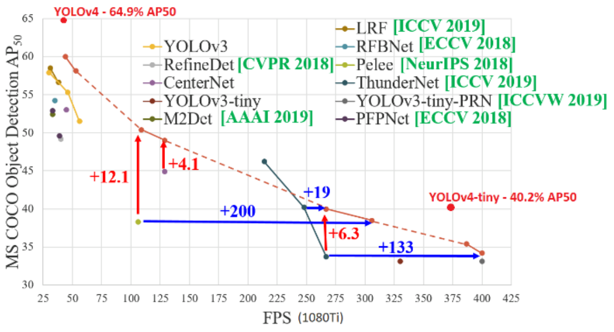


Fig. 3. Detection speed of YOLOv4 by 1080Ti [12]

III. Image Processing Algorithm

In this section, we will introduce the workflow, algorithm, and the architecture of the system.

A. *Overview*

The purpose of this study is to make it safer for passengers to take the bus during the coronavirus. We will design a method to collect the image data from camera on the bus and then analyze the crowdedness of a bus. The algorithm of this project is consisted of detection, tracking, and counting. The data used in the experiment was taken from actual bus cameras. There are two cameras in the bus. One is positioned in the front to toward backwards, and the other is positioned in the center toward the back door. The camera setting position is shown in the **Figure 4**.

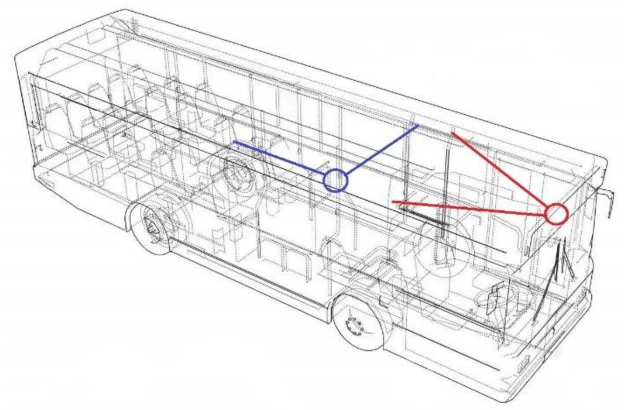


Fig.4. Camera Setup of a bus. The images of getting on the bus will be recorded by the camera marked in blue, and the images of getting off the bus will be recorded by the camera marked in red.

After our analysis of the interior images, we think there are several key points to be solved. First of all, there should be a specific detection method for the detection. Secondly, the crowded bus should be considered in the detection, in which case only the head can be detected. Therefore, we currently plan to identify passengers by detecting their heads. We decide that by counting the number of passengers in front and back doors respectively, the number of passengers can be added or subtracted to infer the real-time number of people in the bus, so as to get the crowdedness. In this process, we also need to determine the statistical method. The above will be described in detail in the next sections.

B. *System Architecture*

This project consists of two cameras on the bus, a cellular module, and the image processing computer. With regard to the cameras, as described above and in Figure 4, after capturing data from the two cameras respectively, the images and videos are analyzed. At present, we are still using ordinary notebook GPU (NVIDIA RTX2060 Max-Q) for image processing and analysis. Our system uses the Linux system as the base operating system, the Python Script as the operating programs. Since the performance of the GPU is acceptable, the deep learning networks currently used are still relatively large now. After the deep learning network model is optimized, it will be loaded into edge computing platforms such as Jetson Nano. Therefore, the theoretical design should be to first capture the image and analyze it in the embedded computer installed in the car, and finally transmit the obtained data back through wireless communication.

TABLE 1
TABLE OF NOTATIONS OF DETECTION

Notation	Description
τ_{iou}	The IoU evaluation index of object detection. Measure the degree to which the prediction box overlaps with the true box.
τ_{score}	The detection result score is used as detection threshold.

C. *Detection*

In the detection process, we use the DarkNet-53 convolutional neural network with residual network. Each convolution part of DarkNet-53 uses the unique DarknetConv2D structure, the L2 regularization is performed during each convolution, and the Batch Normalization Standardization and Leaky ReLU are performed after convolution. Ordinary ReLU sets all negative values to zero, while Leaky ReLU assigns a non-zero slope to all negative values. It looks like as follows:

$$y_i = \begin{cases} x_i & (x_i \geq 0) \\ \frac{x_i}{a_i} & (x_i < 0) \end{cases} \quad (1)$$

In the feature utilization, yolo extracts multiple feature layers for target detection. A total of three feature layers are extracted. The three feature layers are located in different positions of DarkNet-53. After the corresponding convolution processing of the three feature layers, a part is used to output the prediction result corresponding to the feature layer, and the other part is used to perform deconvolution and combine with other feature layers. In this way, the prediction result could be obtained.

In the object detection of this project, we need to detect the humans' heads. Since there is currently no dataset specifically for head detection in vehicles, we use the dataset of the lecture classroom. We use the VOC dataset of heads in lecture [13] from South China University of Technology. We have trained the Part A of dataset on 1800 samples, evaluation on 200 samples, with batch size 4 for freezing layers training and 2 for entire layers training. According to the mAP(Average Precision) test, we adjust parameters τ_{iou} and τ_{score} to get the best accuracy of this model.

D. *Tracking*

The multi object tracking algorithm is based on YOLO object detection algorithm and imported Kalman Filter and Hungarian Algorithm.

The Kalman filter is an efficient recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements. It is used in a wide range of engineering and econometric applications from radar and computer vision to estimation of structural macroeconomic models [14]. The Kalman filter includes two stages: prediction and update. In the prediction stage, the filter uses the estimate of the previous state to make an estimate of the current state. In the update stage, the filter optimizes the predicted value obtained in the prediction state by using the observed value of the current state to obtain a more accurate new estimate.

The calculation of Kalman filter is represented by the following variables:

TABLE 2
TABLE OF NOTATIONS OF KALMAN FILTER ALGORITHM

Notation	Description
$\hat{\mathbf{x}}_{k\|k}$	The posteriori state estimate at time k given observations up to and including at time k;
$\mathbf{P}_{k\|k}$	The posteriori estimate covariance matrix (a measure of the estimated accuracy of the state estimate).
$\mathbf{F}_k$	The state transition model which is applied to the previous state $\mathbf{x}_{k-1}$.
$\mathbf{B}_k$	The control-input model which is applied to the control vector $\mathbf{u}_k$;

Kalman Filter Algorithm [14]:
Predict:
Predicted state estimate:
$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k$$
Predicted estimate covariance:
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$$
Update:
Innovation or measurement pre-fit residual:
$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}$$
Innovation (or pre-fit residual) covariance:
$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$$
Optimal Kalman gain:
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}$$
Updated (a posteriori) state estimate:
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$$
Updated (a posteriori) estimate covariance:
$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}$$

The calculation process of this project is as follows: the image passes through the detector to get the coordinate frame of the human's head, then calculates the position of the center point (x_0, y_0), inputs this (x_0, y_0) to the tracker, which learns and updates, finally gives the prediction, and then repeats the executes to detect all the frames. The workflow of tracking is showed as **Figure 5**.

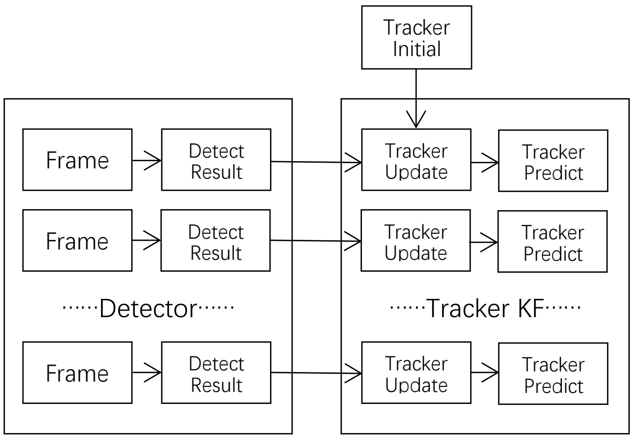


Fig.5. The workflow and the architecture of object tracking.

E. *Counting*

After the people's heads have been detected and tracked, we could get the center points and trackers of passengers' heads. In the process of counting, we will get the coordinates of the previous frame and the next frame of each person's head. We set a line at the entrance to get on and get off as a criterion for getting on and getting off. At the door of getting on, (from the bus outside) when the head crossed the line, the count is increased by one, at the door of getting off (from the bus inside) when the head crossed the line, the count is reduced by one, so as to infer the real-time number of people in the bus.

IV. EXPERIMENTAL EVALUATION

A. *Performance Evaluation*

In this section, we will evaluate the real-time performance of detection and tracking. At present, we have run this algorithm on both ordinary laptop graphics card platform and Jetson Nano platform. The video data used in the evaluation experiment came from the videos of buses' monitors from Kanagawa Chuo Kotsu Co., Ltd. The videos have a resolution of 640x480 and a frame rate of 7.52 frames per second. On the laptop graphics card platform, it can be detected completely in real time, and the frame rate can reach 15~20fps. In Jetson Nano platform, due to the limitations of memory size and hardware, it can run at about 5fps, which can basically meet the minimum detection requirements.

TABLE 3
THE RATE OF FRAME PROCESSING

Platform	DarkNet-53 yolov3	DarkNet-53 tinyv3
Laptop GPU	19.5 fps	31.2 fps
Jetson Nano	3.2 fps	4.9 fps

B. *Model Training and Detection Evaluation*

In this section, we explain the process of model training and the evaluation of the model's detection performance. In model training, we conduct training by importing data, and mainly pay attention to the change of loss value. In the first several epochs, we first froze the training of partial weights, and put more data into the network parameters of the later part of the training, so that both time and resource utilization could be greatly improved. In the process of model training, loss is as high as 6000 at the beginning and finally converges to about 86, and the modified data is not normalized. Overall, the loss curve shows a downward and convergent trend. The curve is showed as Figure 6.

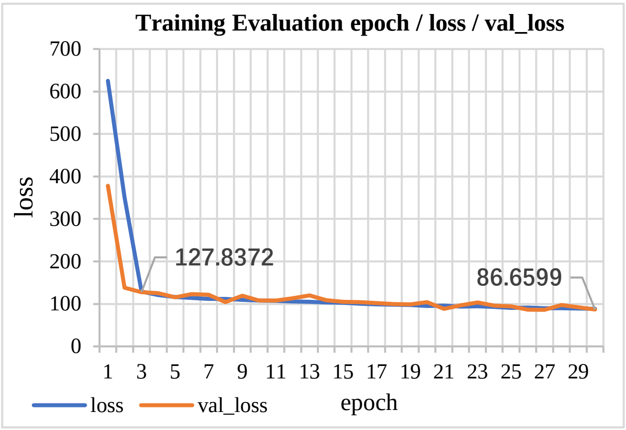


Fig.6. Curve of training evaluation. The figure shows the changes of loss and valuation loss during model training.

After training with 1800 datasets, we use another 200 datasets to test the average precision(AP) of this model and evaluate the detection performance of this model. We combined IoU and threshold to conduct mAP test. For the mAP evaluation method, there are the following evaluation parameters and specific methods. The τ_{iou} is to measure the degree to which the prediction box overlaps with the true box.

$$\mathrm{IoU}(\tau_{iou}) = \frac{S_{\cap}}{S_{\cup}} \tag{2}$$

$S_{\cap}$ is the overlap region between the prediction box and the actual box, and $S_{\cup}$ is the total region occupied by the prediction box and the actual box. Then we evaluated the Precision and Recall of the model. Precision is the ratio of what the classifier considers to be positive classes to what the classifier considers to be positive classes. Recall is the proportion of the part that the classifier considers to be a positive class and is indeed a positive class to all the truly positive classes. We define them by the following abbreviations, and evaluate the AP based on approximated average precision [15].

TABLE 4

TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \tag{3}$$

$$AP = \sum_{k=1}^{N} Precision(k)\Delta Recall(k) \tag{4}$$

Finally, we evaluated the accuracy and stability of the model by testing the same data under multiple groups of participants. We initially started the test with τ_{iou} = 0.3 and τ_{score} = 0.5. Then we fixed the value of τ_{iou} and changed the threshold τ_{score} to detect the change of AP. At the same time, we also fixed the threshold, changed the τ_{iou}, and observed the change trend of AP, as shown in the following tables:

TABLE 5
AP OF DETECTION VS. τ_{score} WHERE τ_{iou} = 0.3

τ_{score}	0.10	0.20	0.30	0.40	0.50	0.60
AP	61.04	60.33	59.61	58.86	57.90	56.53

TABLE 6
AP OF DETECTION VS. τ_{iou} WHERE τ_{score} = 0.3

τ_{iou}	0.10	0.20	0.30	0.40	0.50
AP	56.33	57.45	57.90	58.26	58.47

From Table 5 and Table 6, we can see that different parameters have no great influence on the final detection accuracy. IoU has little influence on the final detection threshold, while confidence has more influence on the threshold. For the crowded bus, in order to avoid the coincidence error of the detection box caused by too small IoU, and finally lead to the detection error, we choose the larger IoU at $\tau_{iou} = 0.3$. At the same time, we can see from the table that the lower the threshold is, the more effective the detection is. Moreover, since we use a special crossline counting method, if there is no detection for static objects, it can be ignored. Therefore, we choose a smaller confidence at $\tau_{score} = 0.1$.

C. *Actual Operation Evaluation*

The project has not yet been run directly on the actual bus, but we used historical video to simulate the actual test as **Figure 7** below.

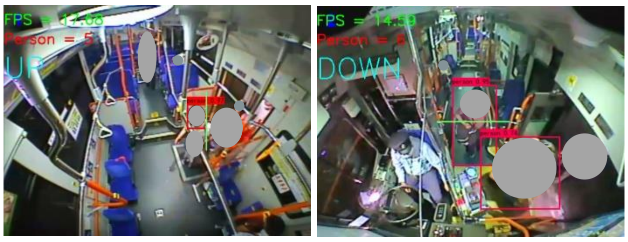


Fig.7. Tests that simulate real bus interior conditions.

In the actual simulation test, we define a tracking rate t_r to evaluate. The tracking rate is the number of frames that are continuously tracked from the beginning of recognition to the end of recognition. We assume that all frames are t_a and that the number of successfully tracked frames is t_s. As shown in the following equation.

$$t_r = \frac{\sum t_s}{\sum t_a} \tag{5}$$

TABLE 7
Tracking Rate t_r

Situation	Getting On	Getting Off
t_r	85.95%	90.54%

In the detection, we found that because the camera of the back door was covered by the handles and railings of the bus, tracking loss often occurred. However, the camera at the front door do not have this problem, so the tracking rate is relatively high. So, we have to avoid that when we draw a decision line. In the video used for detection, a total of 10 people get on the bus and 8 are detected in the end. There are 8 people getting off the bus and 9 are detected in the end. One of them fails the detection because he is wearing a hat. It can be seen from the experiment that the model and the method have certain detection ability.

D. *Future Work*

From the experiments we have written above, we can see that our accuracy is still to be improved, and we have not yet run it on edge computing devices, so we have three areas to work on in the future. On the first hand, we should continue to improve the precision and optimize the detection rate by changing the model structure or optimizing the neural network layers. The second aspect is to train more datasets and special data to improve the recall rate, such as passengers wearing hats and masks. The third aspect is to optimize the running speed of the model by pruning, pooling, distillation, and other methods to reduce the model size so that it could be run on edge computing devices at the real-time detection speed.

V. Conclusion

A method to detect the crowdedness of bus was proposed. We combined the CNN model, Kalman Filter, Hungarian Algorithm, and our unique counting method to realize it. The method currently has quite good accuracy, but we will continue to optimize it according to our expectations. At the same time, we think about whether to detect the behaviors of passengers in the bus other than only counting the number of people. We consider that whether more valuable information could come out. For example, detecting a person who is not in good physical conditions and a suspicious person, etc. Detecting things like left behind, even if they are not identified by humans. Then I think that this method could be applied to a wider range.

Acknowledgment

Thanks for support of Kanagawa Chuo Kotsu Co., Ltd. in the actual route bus demonstration experiment and the video data for the experimental analysis. This research and development work was supported by the MIC/SCOPE #191506001.

REFERENCE

[1] Kanagawa Chuo Kotsu Co., Ltd., "COVID-19 Coronavirus Infection Prevention Strategies," https://www.kanachu.co.jp/dia/news/detail?tbl=4&tid=70 (In Japanese, last viewed on Jan.5th, 2021)

[2] J.-K. Kim, B. Lee, and S. Oh, "Passenger choice models for analysis of impacts of real-time bus information on crowdedness," Transportation Research Record: Journal of the Transportation Research Board, vol. 2112, no. 1, pp. 119–126, 2009.

[3] Rakuten Travel, "Awareness survey on highway buses," https://travel.rakuten.co.jp/mytrip/trend/bus-investigation/ (In Japanese, last viewed on Jan.5th, 2021)

[4] J. Zhang et al., "Analyzing passenger density for public bus: Inference of crowdedness and evaluation of scheduling choices," in Proc. IEEE 17th Int. Conf. ITSC, 2014, pp. 2015–2022.

[5] Yang, T.; Zhang, Y.; Shao, D.; Li, Y. Clustering method for counting passengers getting in a bus with single camera. Opt. Eng. 2010, 49.

[6] Chen, J.; Wen, Q.; Zhuo, C.; Mete, M. Automatic head detection for passenger flow analysis in bus surveillance videos. In Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, Dongguan, China, 28–30 October 2013.

[7] Mukherjee, S.; Saha, B.; Jamal, I.; Leclerc, R.; Ray, N. A novel framework for automatic passenger counting. In Proceedings of the IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September 2011.

[8] Liu, G.; Yin, Z.; Jia, Y.; Xie, Y. Passenger flow estimation based on convolutional neural network in public transportation system. Knowl. Base Syst. 2017, 123, 102–115.

[9] Kenichi Y.; Daisuke M.; Naoto A.; M.; Tomoichi M.; Makoto N.; Congestion degree visualization technology using a monitor camera inside a station. Information Processing Society of Japan Digital Practice Vol.8 No.2, Apr. 2017

[10] Yuma Y.; Satoshi H.; Hirozumi Y.; Teruo H.; Development of Bus Passenger Counter Using LiDAR Sensors. Information Processing Society of Japan Digital Practice 60(3), 934-944, 2019

[11] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv:1804.02767, 2018.

[12] Pjreddie, "darknet," https://github.com/pjreddie/darknet (last viewed on Jan.8th, 2021)

[13] HCIILAB, "SCUT-HEAD-Dataset-Release," https://github.com/HCIILAB/SCUT-HEAD-Dataset-Release(last viewed on Jan.10th, 2021)

[14] Wikipedia, "Kalman Filter," https://en.wikipedia.org/wiki/Kalman_filter (last viewed on Jan.10th, 2021)

[15] Wikipedia, "Precision and recall," https://en.wikipedia.org/wiki/Precision_and_recall(last viewed on Jan.15th, 2021)