Monitoring of Door Status in Public Transit Systems

TING-HSUN CHI, LI-YU CHU, KUAN-YI LEE, and PO-YING TSENG, Department of Electrical Engineering, National Taiwan University, Taiwan

To effectively detect the status of the bus doors, we mix both traditional and machine learning method to extract the key features of the doors.

1 Overview

Our algorithm includes three parts:

1. Machine learning door detection
2. Traditional image preprocessing and feature extraction
3. Feature analysis

For machine learning door detection, we use YOLO as the detection model, and train the model to generate the bounding box of the door. For traditional image processing, we use equalize histogram, Canny edge detection and Hough line detection to detect straight lines. For feature extraction, we detect the distance of the nearest line to the middle of the two doors. For feature analysis, we smooth the data, calculate the slope to determine the opening frames and closing frames. The following sections describes each part in detail, including comparison of different methods and parameters.

2 Machine Learning Door Frame Detection

YOLO is a widely used and technically mature model in the field of deep learning for image recognition. Therefore, we used YOLO v8 to train the model for door frame detection. Considering the need for real-time inference in practical applications, we opted for the smallest pre-trained model (yolov8n.pt) to enhance speed. We found a substantial amount of bus door data on Roboflow, annotated it, and combined it with models from training three video clips, resulting in five integrated datasets for training our model.

2.1 Pre-trained Model Selection

In YOLO, multiple pre-trained models of different sizes are provided. We selected three models for testing.

Model Name	Model Size after Training	Inference Time (Average per Frame)	Miss Rate
yolov8n.pt	6MB	0.9s	10.04%
yolov8m.pt	42MB	3.1s	10.31%
yolov8x.pt	128MB	8.7s	9.85%

Table 1. Comparison of different YOLO models

*Miss Rate = Number of frames with door frames not detected / Number of frames with door frames

Although the official data shows a significant improvement in accuracy using yolov8x.pt, due to the relatively simple task of door frame detection in our case, there is no significant difference in accuracy among the three models. Moreover, our method only requires the model to provide the average object bounding box range for all door frames, thus we can

Authors' Contact Information: Ting-Hsun Chi, b10901184@ntu.edu.tw; Li-Yu Chu, b10901013@ntu.edu.tw; Kuan-Yi Lee, b10901091@ntu.edu.tw; Po-Ying Tseng, b10901074@ntu.edu.tw, Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

tolerate a small amount of error. Therefore, after considering, we decided to prioritize inference time and chose the smallest pre-trained model, yolov8n.pt.

2.2 Training Dataset Processing

Initially, we encountered issues where the model could not detect fisheye door frames or door frames not at the top of the screen. Therefore, we performed two different augmentations on the dataset. First, we rotated part of the dataset by 90 to 270 degrees and shrunk some of the images, leaving black edges. The new model could then correctly interpret the fisheye door frames.

Model Name	Inference Time (Average per Frame)	Detected Frames (09.mp4)
Original Model	0.78s	4
Model after Data Augmentation	0.9s	36

Table 2. Performance comparison before and after data augmentation

It can be seen that the new model has significantly improved accuracy.

2.3 Inference

For inference, we extracted one frame for every two frames to speed up the process (modifiable in practical use), and we returned the door's position and frame (cls[0] for close, cls[1] for open).

To avoid misdetecting non-door regions as the door, we used the median coordinates of all OPEN signal bounding boxes. This approach ensures that a few misdetected signals do not affect the overall door frame detection. We only use the OPEN signal and not the CLOSE signal because our tests showed that during the door's opening/closing events, the door is in the OPEN state, thus the OPEN bounding box more accurately encompasses the line detection range.

3 Image Preprocessing

After we get the rough frame of where the doors are, we need to get the edges of the doors to sketch the shape of them. We notice that the most obvious feature that can separate doors from other objects (people, animals, etc.) is that doors are mostly made of straight lines. We develop a novel way to extract straight lines that represent the shape of doors with high confidence. The method is as follows:

1. Turn each image into grayscale, and equalize the histogram
2. Apply Gaussian blur and Canny edge detection.
3. Apply Hough line detection to detect straight lines.

These methods combined can sketch the shape of doors. We explain each method below, including comparisons and parameter tuning.

3.1 Equalize Histogram

We do equalize histogram only in the bounding box detected by YOLO (see previous section). This technique let the line detection algorithms to be able to detect the shape of the doors under severe conditions, such as night time. 1a and 1c shows the line detection result with/without equalizing histogram. It can be seen that when it is dark outside, the straight lines of the door edges cannot be detected without equalizing histogram.

(a) (b) (c)

Fig. 1. Comparison between (a) line detection with equalize histogram in the bounding box (b) line detection with equalize histogram on the whole image (c) line detection without equalize histogram. Blue lines indicate the bounding box and the middle of the bounding box, green line indicate the line detection result, red line indicate the line closest to the middle of the doors. The detection result in (a) is the best of the three.

(a) (b)

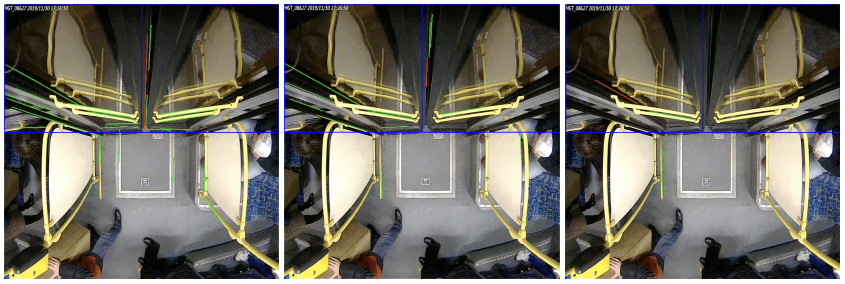
Fig. 2. Comparison between (a) line detection with equalize histogram with Gaussian blur (b) line detection without Gaussian blur. The detection result in (a) is less noisy.

We choose to only consider the pixels inside the bounding box so the difference between the background and the doors inside our region of interest can be more distinguishable. 1a and 1b shows the comparison of equalizing the whole image and only equalizing the bounding box. It can be seen that equalizing the whole image is less effective in detecting straight lines in the dark.

3.2 Gaussian Blur and Canny Edge Detection

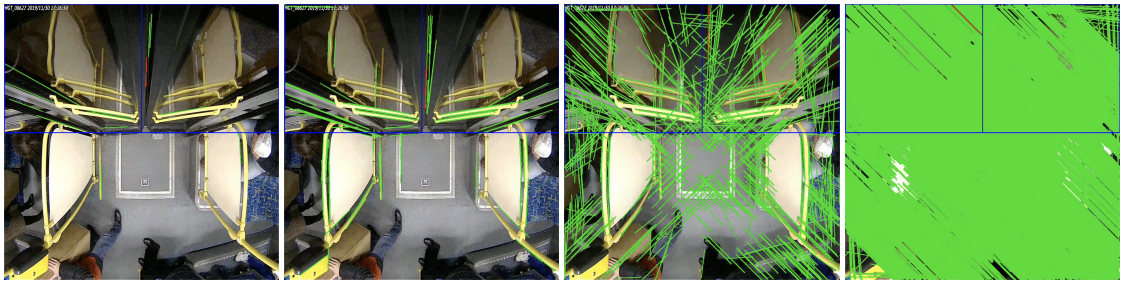
The Gaussian blur is an important step after equalizing histogram. This is because when we equalize histogram, some small lines that we don't want will be enhanced, such as the the separation between the bricks on the floor in 2. These unwanted lines can be cleaned by Gaussian blur. 2 compares the line detection with/without Gaussian blur, more unwanted lines are detected without Gaussian blur.

The kernel size of Gaussian blur is also important, in our algorithm, we use kernel size (5,5) after comparing different kernel sizes. 3 compares different kernel sizes. It can be seen that the kernel size (5,5) best rule out unwanted edges while preserving the edges of doors.



(a) (b) (c)

Fig. 3. Comparison between (a) 3×3 (b) 5×5 and (c) 7×7 kernel. The detection result in (b) is the best of the three.



(a) (b) (c) (d)

Fig. 4. Comparison between (a) Canny edge detector (b) Sobel operator (c) difference of Gaussian, and (d) Prewitt operator. The detection result in (a) is the best of the four, others have unwanted lines.

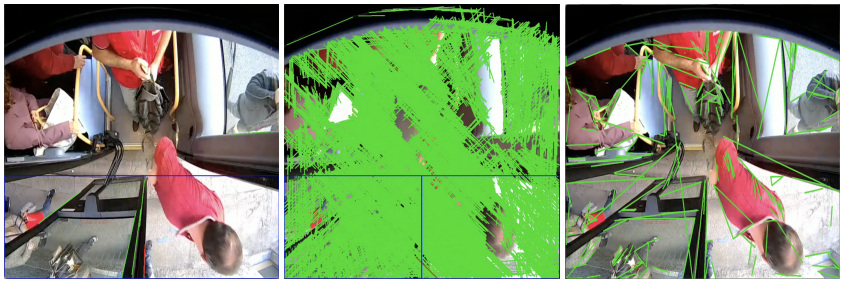
Next, we utilize Canny edge detection before applying line detection algorithm. This step is commonly used in line detection pipeline, as Canny edge detection can roughly extract edge features, which is often similar to line features, so the line detection algorithm can perform better. We have also tried different edge detection algorithms, including Canny edge detection, Sobel operator, difference of Gaussian, and Prewitt operator. 4 shows the comparison of line detection with different edge detection methods. Canny edge detection can perform better than other methods.

3.3 Hough Line Detection

To detect the shape and frame of doors, we attempt different methods, including polygon approximation of contour and Hough line detection. We eventually decide to use Hough line detection due to its robustness.

We first try the polygon approximation of contour method. We extract the contour of the image, and use the `cv2.approxPolyDP` to get an approximate polygon of the shape. If the approximated contour has 4 vertices, we consider it an edge that is related to a door-like object. This method can extract straight line features, but the quality of the results vary a lot, so we choose other method.

Then, we take a stab at Hough line detection method. We compare `cv2.HoughLines` and `cv2.HoughLinesP` and found out that `cv2.HoughLines` usually yields a mess (too many lines detected) while `cv2.HoughLinesP` performs



(a) (b) (c)

Fig. 5. Comparison between (a) cv2.HoughLinesP, (b) cv2.HoughLines, and (c) cv2.approxPolyDP of contour. The detection result in (a) is the best of the three, others have unwanted lines.

reasonably well. 5 compares the detection result of `cv2.HoughLinesP`, `cv2.HoughLines`, and `cv2.approxPolyDP`. It can be seen that the outcome of `cv2.HoughLinesP` is more stable than `cv2.approxPolyDP`.

4 Feature Extraction

After getting the lines that might related to the doors, we need to find the key feature that resemble the opening and the closing of doors. The key feature needs to satisfy the following property:

(1) Enough variation between the states of the door, i.e. opening and closing
(2) Remaining stable at other times

4.1 Different Features

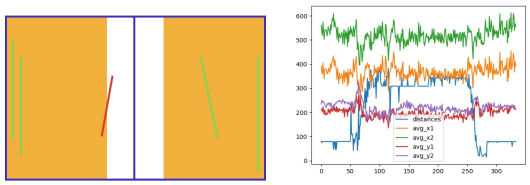
We attempt two methods:

(1) The average coordinates of the lines detected at each frame
(2) The distance between the door gap and the line closest to the door gap

as shown in 6a. 6b shows the chart of the two features for video 3 in training set. Here, we define the ”distance” as follows:

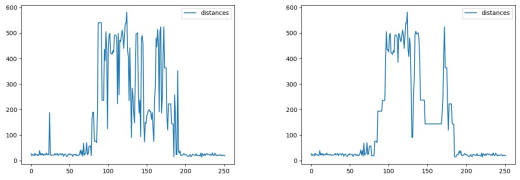
$$d = \begin{cases} \sqrt{(x_1 - x_{1,gap})^2 + (x_2 - x_{2,gap})^2}, & |x_{1,box} - x_{2,box}| \geq |y_{1,box} - y_{2,box}| \\ \sqrt{(y_1 - y_{1,gap})^2 + (y_2 - y_{2,gap})^2}, & |x_{1,box} - x_{2,box}| \leq |y_{1,box} - y_{2,box}| \end{cases}$$

The two condition is derived from the fact that the bounding box is always a horizontal rectangle when the door is vertical, and a vertical rectangle when the door is horizontal. It can be seen that the distance between the door gap and the line closest to the door gap better matches the door open status and close status, so the transition would be the opening and closing. It is not hard to find out that the second method is exactly how human estimate if the door is open or closed: we see if the edge of the doors separate large enough from the middle, meaning the gap between the doors is large enough.



(a) (b)

Fig. 6. (a) a toy figure of the closest line to the door gap. The blue line in the middle is the door gap, the orange rectangles are the doors (opening/closing), and the red line is the closest line to the door gap. (b) different possible feature for public test video 3, the average of x1, x2, x3, and the self-defined closest line's distance. The self-defined distance changes with the door's status, so it is a better feature.



(a) (b)

Fig. 7. (a) the distance feature without the selection rules (b) with the rules.

4.2 Advanced Feature Selection

The line detection result from Section 3 can still be noisy to get exactly the edge of the doors as the closest lines. Therefore, we add several rules to prevent the feature extraction to take unwanted lines as feature. We set two rules:

1. The distance of the current closest line cannot be too far from the closest line in previous frame.
2. The angle between the line and the direction of the door gap cannot be too large.

If the current closest line does not meet the above rules, we take the distance of the previous frame as the distance of the current frame. The first rule prevent the algorithm to accidentally choose the lines on other objects than the door edge, such as a box held by a passenger. The second rule prevents the algorithm to view the top of the door frame and the bottom of the door frame, or the handle of the doors, as a valid feature.

7 shows the comparison of feature data with/without the rules. It can be seen that without the first rule, the distance of the closest line might vary a lot between continuous frames, yielding uninterpretable results. Without the second rule, the algorithm might choose the top/bottom/handle as the closest line, which is not what we are looking for.

5 Feature Analysis

With the distance features, we can observe some consistent properties at the timing of opening and closing. Therefore, we develop an algorithm to predict the frames from the observation and some analysis of the properties.

5.1 Feature Processing

While 7 still shows enormous fluctuations, we apply three functions sequentially to perform noise reduction and smoothing: `scipy.ndimage.gaussian_filter1d`, `scipy.signal.savgol_filter` and `ema`.

1. Gaussian filter: a smoothing technique that uses a Gaussian function to perform weighted averaging. The output of the Gaussian filter is obtained by convolving the input signal with a Gaussian kernel. This process assigns more weight to the central values and less to the values further away from the center, effectively reducing high-frequency noise and smoothing the signal.
2. Savitzky-Golay Filter: a digital filter that applies a polynomial smoothing method. It fits successive subsets of adjacent data points with a low-degree polynomial by the method of linear least squares, which is useful for preserving the shape of the signal, such as peaks and troughs, while reducing noise.
3. Exponential Moving Average: a type of infinite impulse response filter that applies weighting factors which decrease exponentially:
$$f(t) = \alpha \cdot x_t + (1 - \alpha) \cdot f(t - 1)$$
It adds an additional layer of smoothing and is more responsive to recent changes in the data compared to a simple moving average, making it useful for applications that require a more dynamic response to changes.

By applying these filters in sequence, we effectively reduce noise and achieve a smooth signal that retains essential features and adapts to recent changes in the data. 8 shows some of the results.

5.2 Properties of the feature

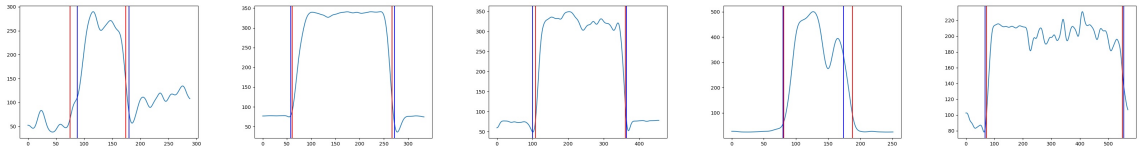
(1) After comparing with the ground truth and the distance features, we found that when the distance increases/decreases significantly, it is mostly the timing of door opening/closing.
(2) The distances of two states of opening and closing are different, and thus the states can be distinguished by roughly calculating their average distances respectively.
(3) Although there are still some unexpected ripples varies quickly, they usually do not sustain for many frames after the processing of line detection. Therefore, our algorithm can prevent it from disturbing.

5.3 Algorithm

Our algorithm mainly consists of two iterations. The first iteration is used to simply identify the time interval of opening door and closing door, and the second iteration is actually used to detect the objective frame. Because the case of detecting door opening and closing is opposite, we omit it in the following part of explaining algorithm.

In the first iteration, we computed the average of the whole process as the standard of separating two states of door, and then search for the interval where the distance is monotonically increasing, and note that the process must cross the average. We also set a threshold to deal with the ripples mentioned above. If some ripples accidentally cross the average, we will ignore them if the interval between crossing the average two times is shorter than the threshold. In this way, we get the rough timing of opening, and set the endpoint of the intervals as starting open and ending open. Because the objective in this iteration is only to get the interval when the door is opened or closed, despite the method may detect wrong timing, it still give an acceptable separation of two states.

After that, we derive the average distance of two states, denoted as closed state distance and opened state distance, and set the average of the two distances as the new standard instead of that in the first iteration, and start the second iteration.



(a) (b) (c) (d) (e)

Fig. 8. The distance feature of 01.mp4, 03.mp4, 05.mp4, 07.mp4, 09.mp4. Blue straight lines are ground truth, red straight lines are guessed frames.

In the second iteration, the way of detecting the timing of door opening only has some subtle difference compared to the first part. We add some conditions when determining the endpoints of the interval to predict more precisely. 1. When selecting the endpoint of start opening, if the distance of the frame is close to the closed state distance, then stop scanning other frames, and for the other cases is in similar way. 2. After choosing endpoints by the first condition, we modify the endpoints by restricting the slope. It is reasonable to expect that the slope will be some specific value but not close to zero or some large number at the moment when door starts opening. Consequently, we set a range and make sure that the slope of modified endpoints is in the range. The final step is to generate the guessed frame by choosing the frame between the two endpoints of start open and end open. This is how we calculate the guessed frame:

$$guessed\ open\ frame = \frac{7}{8} \times start\ open\ frame + \frac{1}{8} \times end\ open\ frame$$
$$guessed\ close\ frame = \frac{1}{8} \times start\ close\ frame + \frac{7}{8} \times end\ close\ frame$$

6 Final Result

Using the above algorithm and data analysis method, we are able to reach perfect F1 score in public test cases. 8 shows the plot of our main feature (distance), our guessed frames, and the ground truth frames. The value of distance varies according to the status of the door, and our guessed frames match the ground truth frames. For the private test set, many problems occur. We analyze the wrong videos one by one:

02.mp4

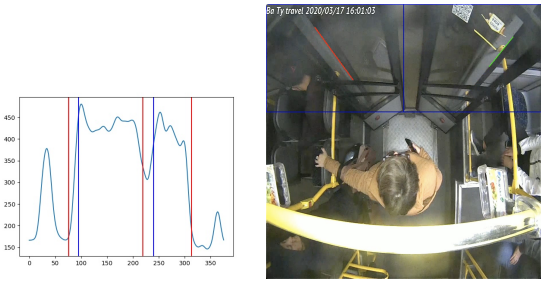
The problem in this video is the detection of lines are not stable. Our histogram equalization method did not do enough job for the Hough line detection to detect the gap of the doors, so there are some times when the door is closed, the only lines detected are the outer door frame, then our algorithm will misinterpret it as an open door, as shown in 9.

06.mp4

The problem in this video is a false-alarm. 10a shows the distance plot, the plateau should be interpret as a closed door, but our algorithm determines it as open. It shows that when the angle of the camera to the door is really strange, causing the distance between open and close varies only a little, our algorithm might sometimes lead to false-alarm.

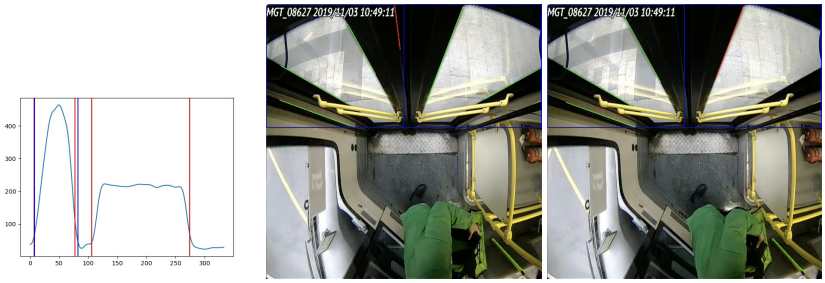
08.mp4

The problem in this video is a delayed-detection. 11 shows the distance plot and a snapshot. When the door is opening, the closest line detection abandons the detected door edge since the difference of distance exceeds the threshold in



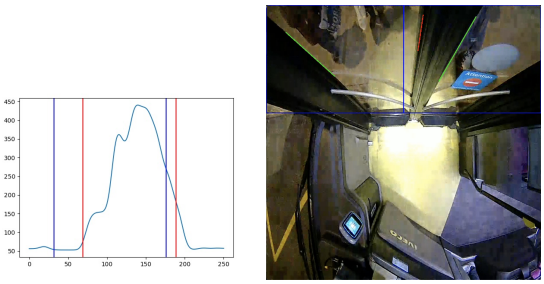
(a) (b)

Fig. 9. 02.mp4: (a) Distance (b) The miss-detection of lines.



(a) (b) (c)

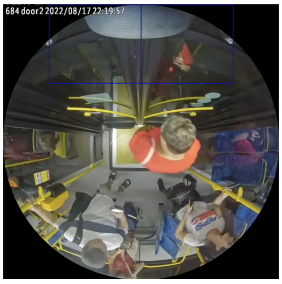
Fig. 10. 06.mp4: The distance difference between (b) and (c) causes the plateau



(a) (b)

Fig. 11. 08.mp4: The door gap is not in the middle of the bounding box.

Section 4.2. This is due to the false marking of the door gap. We mark the door gap as the middle of the bounding box, but in this video, due to the strange angle of the camera, the door gap is not in the middle of the bounding box, so when the closest line skip from the left door to the right door, it exceeds the threshold, leading to a delayed-detection.



(a)

Fig. 12. 10.mp4: A wrong bounding box detected by YOLO.

10.mp4

The problem in this video is a wrong bounding box. The fish-eye camera makes YOLO to detect a smaller bounding box, as shown in 12. When running our algorithm, there are no lines that totally lies inside the bounding box, leading to miss-detection.

To sum up, our algorithm is based on the intuition for human to detect the door status, but this method might be sensitive to different camera angle. For future improvement, the line detection can be done on the frame-by-frame difference map, so moving objects can be detected more easily. The robustness of our YOLO model can be further improved by data augmentation through distortion.