Computer Vision System for Railway Level Crossing Monitoring

MARSOT Mathieu

Abstract

In this report, a computer vision system to monitor a Railway Level Crossing is presented. The purpose of this monitoring is to detect intrusive objects in the image. It is not a classical CCTV problem as it is hard to have a knowledge of the background image because the chronology of the images cannot be assumed. To overcome this problem, two segmentation algorithms are combined: an automatic histogram thresholding algorithm and an edge detection algorithm. Once the objects are detected they are then labelled and classified using OpenCV functions and calculating their intersections to some predefined zones of interest.

Introduction

The major requirement of the program is to monitor a railway level crossing by detecting 5 different situations. The algorithm should be able to know when the railway track is not clear (event 1), when a vehicle is entering the crossing (event 2), leaving the crossing (event 3), when the safety barrier is deployed (event 4) and when a train is on the track (event 5). Of course, it should be able to detect multiple events occurring at the same time as well as the absence of event (event 0).

The CCTV images are taken from a fixed position and only daylight images are considered, however the chronological order of the images cannot be assumed and therefore most of the existing video background/foreground extraction techniques are not as efficient as they could be.

The proposed algorithm tries to extract a binary foreground mask from a given image based on a reference background, the reference background is fixed and will not change during the whole observation. It then classifies the different components of the mask in the 3 different zones of interest. Finally, it looks for the safety barrier to see if it is deployed.

The zones of interest are predefined but if the user wishes to redefine them, a user interface is provided in the algorithm. The choice of the zones of interest will not be discussed in the report but it has a big impact on the quality of the detection.

Foreground extraction

The foreground extraction is based on the difference image between the current image and a reference background image.

First, an automatic histogram thresholding is used to get the foreground. The method looks for the turning points of the histogram of the difference image as described in [1]. However, in [1] it is assumed that the difference histogram has a peak at 0 as a big enough proportion of the image hasn't changed. It is not always true in this case because the reference image is not changing, and the global illumination is changing so there is always a small offset and the peak is rarely at 0. To tackle this issue, the algorithm looks for the

highest peak close to 0 and use it as starting peak.

Then the foreground mask is refined by removing the shadows of the detected objects, the shadow removal method is presented in [2] and is testing the value of the quotient of the foreground pixel value (F) on the background pixel value (B) against a given threshold. Here, if $1 < \frac{F}{B} < 1.6$, the foreground pixel is considered as a shadow and removed.

This method works well if the reference image illumination is close to the actual background image (Figure 2).

Figure 2: Foreground mask obtained via histogram thresholding

However, this method tends to fail on the grass part of the image when the illumination of the image differs too much from the illumination of the reference image (Figure 3) or when a train passes by. Indeed, trains are modifying the image statistics so much that the histogram is not reliable anymore.

Figure 3: Foreground mask with illumination problems

To tackle this illumination / train problem, another foreground mask is extracted, this time using canny edge detection on the difference image and then filling the edges with a constraint run length algorithm which fill the gaps of less than X/Y black pixels between two white pixels on the x/y-axis (Figure 4).

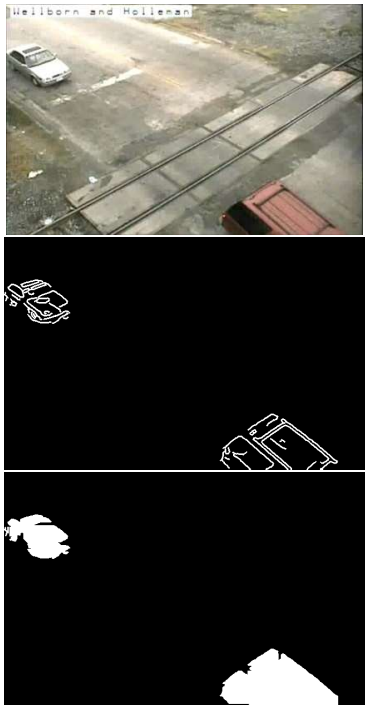
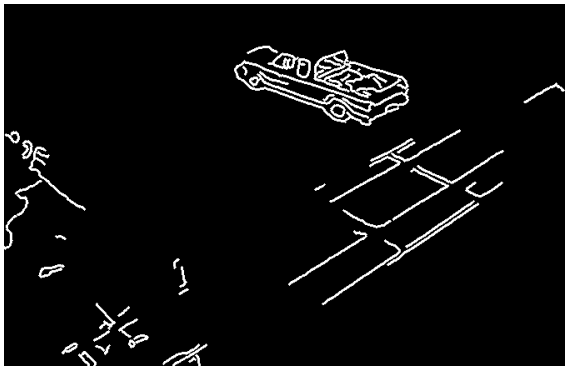
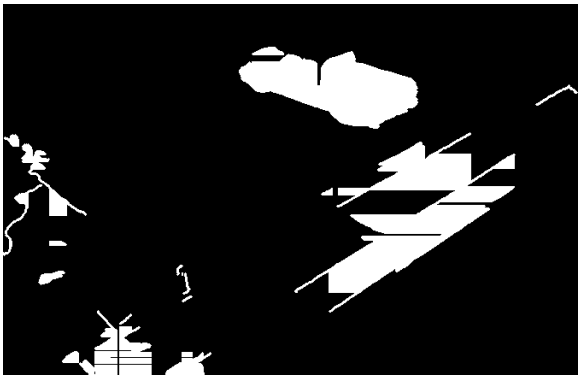


Figure 4: Canny mask extraction (canny edge in the middle, filled mask on the bottom)

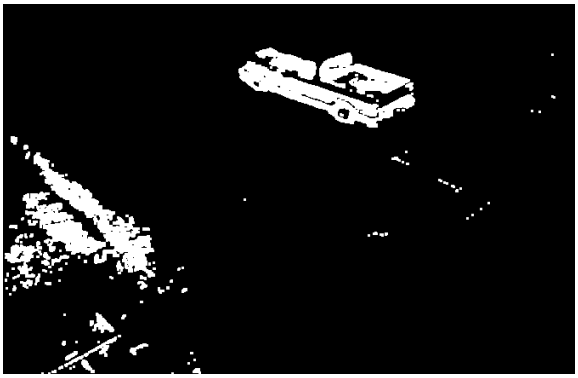
This canny segmentation has the advantage of being consistent for trains. In addition, when the illumination problem occurs, the unwanted foreground is mostly located on the railway and not in the grass parts of the image (Figure 5). The fusion of the two masks (logical and) therefore gives a more robust foreground mask.



(a) Canny edge detection



(b) Canny filled via CRLA



(c) Histogram foreground

Figure 5: Foreground extraction comparison

Finally, a morphological closing operation is done on the fused mask to fill small holes in the detected objects and prepare them for the labelling.

This foreground extraction method gives good result but if the illumination difference between the reference image and the current frame reaches a certain point, the histogram method completely fails, and the entire image is considered as foreground. Therefore, there will be a lot of noise on the track. The shadows of the barrier and of a street light are also not handled and are often detected as intrusive objects.

Labelling and classification of events

After the segmentation. The small objects are removed from the masks. To do so, the *findContours()* function of OpenCV is used on the foreground mask and then the *approxPolyDp()* function is used to transform each contour in a polygon and then the area of each polygon is computed using the OpenCV *contourArea()* function. If the area of a contour is below a threshold (here, 400 pixel²), the object is removed.

Once the small objects have been removed, the *connectedComponents()* function of OpenCV is used to label each object (Figure 6) and the intersection of the obtained objects with the zones of interest are computed. Each object is then classified as entering, leaving, or on track depending on the proportion of the object in each zone.

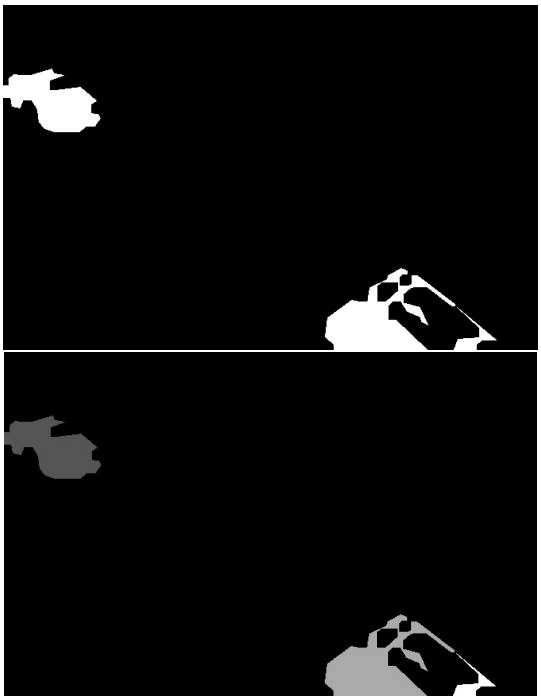


Figure 6: left = mask resulting of the foreground extraction, right = labelled mask

If a big object (bigger than 60,000 pixel²) is detected, the zones of interest are modified because it may be a train, this is required to prevent false detections when a train is on the railway. However, it can backfire if a big truck is on the track as it may consider it as a train.

As the foreground mask is the fusion of two masks, some vehicles can be split into two or more parts and are therefore identified in multiple zones. A morphological closing is applied to reduce this problem but on the other hand, close objects can also fuse into a single object and will therefore be identified only once.

Barrier detection

The foreground extraction and the classification of the detected objects allows the algorithm to detect entering, leaving and on track vehicles as well as trains. However, it is not dealing with the deployment of the safety barrier.

The detection of the barrier is based on the probabilistic Hough line transform (OpenCV *HoughLinesP()*) and detect if there is a big enough line in a predefined zone. If there is a line, an additional test on the mean value of the red channel of the line is done to detect is it is a shadow or not. If the mean value is less than a given threshold (here 150), the line is considered as a shadow otherwise it is the barrier. On some images, the barrier appears darker than usual and the algorithm may consider a deployed barrier as a shadow.

Failed attempts

Region growing was tested to improve the quality of the detected objects, hoping that it would divide the fused ones and complete the split ones. However, grey vehicles often have a colour to close from the road, so the region were growing out of the car and almost everything in the image. In addition, cars

almost always have at least two regions: one for the windscreen and one for the car body and the train have a lot of regions so it was an inefficient approach.

Self-quotient image (presented in [3]) were tested to reduce the noise of the canny foreground mask. However, the quality of the edges of the vehicles in the image are not good enough and the edges were sometimes going undetected in the SQI, it did reduce the noise on the railway however it was missing the dark vehicles.

Normalized RGB images [4] were tested to prevent uneven illumination, however as almost every vehicle are shades of grey it was losing too much information and only the red vehicles were well detected.

Possible improvements

The algorithm works well if the reference background image illumination is close from the examined frame. It would therefore be nice to find a method to compute the reflectance of the background image and therefore to be able to minimize the effects of an uneven illumination.

The reflection of light on the rails makes a lot of false positives and is the only major problem that has not been addressed. Using a mask to cover them or trying to classify them according to their orientation does not work well as it was deteriorating the quality of on track vehicles detection. Maybe a kind of pattern recognition in the canny could classify some edges as belonging to the railway however no existing algorithm or ideas were found for this task.

References

1. Hsin-Yi Wang, Li-Hung Wang, Chung-Bin Wu, "An efficient background extraction and object segmentation algorithm for realtime applications", *Circuits and Systems (APCCAS) 2012 IEEE Asia Pacific Conference on*, pp. 659-662, 2012.

2. A. Bevilacqua, "Effective shadow detection in traffic monitoring applications", *J. WSCG*, vol. 11, no. 1, pp. 57-64, Feb. 2004.

3. Gopalan R, Jacobs D. Comparing and combining lighting insensitive approaches for face recognition. Comput Vis Image Underst. 2010;114(1):135–145.

4. http://aishack.in/tutorials/normalized-rgb/