A deep learning approach to trespassing detection using video surveillance data

Muzammil Bashir, Elke A. Rundensteiner, Ramoza Ahsan

Department of Computer Science, Worcester Polytechnic Institute, United States

Abstract

While railroad trespassing is a dangerous activity with significant security and safety risks, regular patrolling of potential trespassing sites is infeasible due to exceedingly high resource demands and personnel costs. There is thus a need to design an automated trespass detection and early warning prediction tool leveraging state-of-the-art machine learning techniques. Leveraging video surveillance through security cameras, this thesis designs a novel approach called ARTS (Automated Railway Trespassing detection System) that tackles the problem of detecting trespassing activity. In particular, we adopt a CNN-based deep learning architecture (Faster-RCNN) as the core component of our solution. However, these deep learning-based methods, while effective, are known to be computationally expensive and time consuming, especially when applied to a large amount of surveillance data. Given the sparsity of railroad trespassing activity, we design a dual-stage deep learning architecture composed of an inexpensive prefiltering stage for activity detection followed by a high fidelity trespass detection stage for robust classification. The former is responsible for filtering out frames that show little to no activity, this way reducing the amount of data to be processed by the later more compute-intensive stage which adopts state-of-the-art Faster-RCNN to ensure effective classification of trespassing activity. The resulting dual-stage architecture ARTS represents a flexible solution capable of trading-off performance and computational time. We demonstrate the efficacy of our approach on a public domain surveillance dataset.

Keywords: Trespassing detection, Railroad security, Deep learning, Video surveillance, Deep Convolutional Neural Networks, Background subtraction, Computer vision

1. Introduction

Railroad trespassing is a widely discussed issue in railroad security[]. From 2006 to 2015, 2717 deaths and 9595 injuries have been reported to be a direct result of trespassing activity[]. This amounts to 3.37 casualties every day. According to Federal Railroad Administration (FRA), there are around 210,000 railway crossings in USA and around 61% of them are exposed to potential trespassing activity. Each of this site poses a risk to both trespassers as well as train. In most cases, collision with a train proves to be fatal for the trespasser. Aside from human costs, these accidents, whether fatal or not, are exceeding expensive. Property damage, emergency services, safety investigations, insurance, legal and delay costs may account for hundreds of thousands up to millions of dollars per accident[].

A considerable amount of research has been conducted to understand and find solutions to this problem[]. cite paper that proposes manual soln. One solution is to set up a surveillance network of CCTV cameras and employ human analysts to review the video feed on 24×7 basis. Video surveillance data can be transformed to infer trespassing statistics. This can be useful to determine potential trespassing location and time for more efficient resource utilization i.e. police officers or relevant personnel (such as social workers) can be sent to potential sites only as per need basis. However, one major limitation of this solution is the sheer overwhelming requirement of large number of trained human analysts. These analysts have to review tens of hours of CCTV data from hundreds of cameras. Manual processing this "big data" is simply non-practical and infeasible in this age of Artificial Intelligence (AI) and Machine Learning (ML). Further, human analysis has additional drawback of subjectivity and unreliability due to dull and mundane nature of task cite a paper that says this is dull work.

Due to above mentioned reasons, bringing automation and artificial intelligence to any trespassing prevention solution is of vital importance. Trespassing detection indeed serves as the first step towards any auto-

mated AI-based trespassing prevention solution. A reliable automated trespassing detection system will not only provides detection in a timely manner but will also allows us to develop advanced analytics by studying trespassing patterns over time. For example, analysis over a period of three months may reveal that a group of children like to play football during the evening time. Certain locations might see increased trespassing during the morning and/or evening times because people returning home from jobs may want to take a short-cut. Other locations such as underpasses and bridges may provide a preferred meeting location for drug addicts. Use of these advanced "big data analytics" can help us make better predictions and thus assist in reducing trespassing activity substantially.

1.1. Problem:

Given an input surveillance video, the problem of trespassing detection is to classify whether each frame has human trespassing activity or not. In order to keep it simple, we define trespasser as a human spotted near a railway line. Notice that anyone within the camera field of view shall be considered a trespasser by our currently proposed solution.

1.2. Goals:

Although in problem definition, we formulate the problem we tackle as classifying each frame as trespassing or not; we have a more ambitious goal. We not only want to predict the label but also want to do so in a time-efficient manner. We notice that railroad surveillance video is sparse in terms of trespassing activity i.e. in a given 24 hours of railroad surveillance video, most of the video shows no trespassing activity. We aim to leverage this property of sparseness to reduce the processing time.

Further, we postulate that the detection performance and speed (of detection) are two opposite goals. Generally, if one wishes to improve the speed, they will have to sacrifice accuracy and vice versa. Therefore, we are interested in developing a flexible solution that is capable of trading-off performance versus computational time.

2. Related Work

Since our work is at the cross section of railroad trespassing related safety and use of computer vision in trespassing detection, therefore we shall review the relevant work related to both.

2.1. Railroad trespassing related safety

Considerable efforts have been made to understand the risks associated with trespassing activity and developing strategies to reduce those risks. Caird et al.[1] developed a taxonomy of human factors that contribute towards the unsafe human activity. The taxonomy groups common accident contributors in 6 categories: unsafe actions, individual differences, train visibility, passive signs, active warning systems and physical constraints. As opposed to Caird et al.[1], Sussman and Raslear[2] indicated intentional, distraction-caused or other (visibility issues or driver confusion) reasons for accidents. All of these issues require different approaches to solve the problem. However, those approaches can broadly be classified as engineering, education and enforcement-based.

Apart from studying the strategies to reduce the risk, efforts have been made to quantify the frequency and severity of accidents. US Department of Transportation (DOT) accident prediction model is the most widely used method to predict expected no. of collisions per year, at a given crossing site [3, 4]. It is based on different parameters related to traffic volume, train time table and previous accident history [5]. This method has been compared to Transport Canada Accident Model [6] by Chaudhary et al.[7]. Chaudhary et al. reports that overall US DOT model predicts the yearly number of accidents more accurately. However, in cases where the crossing had an accident history, the Transport Canada model outperformed US DOT model. As a conclusion, they suggested to adapt the Transport Canada model to US crossing data and use it to rank the most dangerous crossings.

Different engineering strategies used to prevent collision can be broadly classified as sealed corridors, obstacle detection and traffic channelization[5]. The concept of sealed corridors presents an ideal solution to the problem, however, high costs and reduced access renders it less attractive. Obstacle detection refers to detecting the presence of person or vehicle on tracks and communicating it to the approaching train[8]. It provides a cost-effective and feasible way to reduce collision risk, however the main challenge is the short reaction time available to bring the train to stop. Glover[8] suggests that there may be limited reduction in severity of a collision because train may still collide. However, Hall[9] argues that obstacle detection may still be beneficial as it might give invaluable time to decelerate the train sufficiently to save human life. Traffic channelization is another effective strategy which separates the traffic flow from rail tracks. Federal Railroad Administration research shows positive results and suggests

that it discourages risky driving behaviour around the crossing[10].

2.2. Artificial intelligence

Very little prior work has been done to use computer vision for railroad trespassing safety. Shah et. al[11] employed the gradient and color-based background subtraction to detect moving objects. They use color, motion and size-based features to track those objects. Tracked objects are classified into people, group of people or vehicle. Salmane et al.[12] proposed a multi-stage system that uses frame-based background subtraction for moving object detection in the first stage. However, the moving objects are not discriminated into train, vehicle or persons if the system is to be used for trespassing detection. Further, as noticed by Zhang et al.[13], the system is expected to run in real time, however no information regarding the speed of the algorithm's performance is reported. Zhang et al.[13] developed a practical near-miss trespassing detection system. Their main focus is on detection of near-miss events on gated crossings rather than general trespassing detection. They initially identify time interval during which gates are closed and then detect the moving objects using background subtraction. However, they use no. of pixels as classification metric to distinguish between train and other objects (people and vehicles). This strategy works for their test data but may fail if the camera is located away from train, since it is based on the assumption that train will always constitute most of the moving pixels.

One of the major drawback common to all the above mentioned systems is that they simply use background subtraction for object detection. Further, the method used is mostly frame subtraction-based which is known to generate noisy output cite. Only Shah et al.[11] uses some hand-crafted features to classify the object type. However, recent research suggests that deep-learning based approaches outperform these hand-crafted features based approaches[14]. This limitation in literature motivates us to develop a deep learning based trespassing detection system that can reliably detect human trespassing activity.

3. Methodology

3.1. Problem formulation

Given an input surveillance video containing N frames, we want to produce a binary time series of same length N such that each index i predicts the labels y_i of

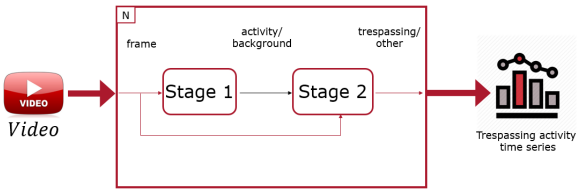

Figure 1: Trespassing detection pipeline

corresponding frame f_i . Human trespassing label is assigned to positive class (1) and "other activity" label is assigned to negative class (0). Since, each prediction depends only on corresponding frame f_i, our problem boils down to determining a function D such that

$$D(f_i) = \hat{y}_i$$

This function D has parameters θ such that $D(f_i; \theta) = \hat{y}_i$. The aim is to find a θ^* such that $D(f_i; \theta^*) \rightarrow y_i$ where y_i is the ground truth label corresponding to f_i. The ground truth label has the following definition:

$$y = \begin{cases} 1, & \text{if } f_i \text{ has trespassing activity} \\ 0, & \text{otherwise} \end{cases}$$

We define the trespassing activity as the presence of at least one person in the frame.

3.2. Pipeline overview

In order to tackle this problem, we propose a two-stage trespassing detection model. Figure 1 shows the block diagram of our system implementing trespassing detection framework. In the first stage, we decide whether a particular frame has activity or not. If it turns out that the given frame has no activity, then it is classified as background frame. No further action needs to be taken for this frame. On the other hand, if it shows activity, then the next step (stage 2) is to investigate whether it can be classified as human trespassing activity or not. Input to our pipeline is a video and each frame is processed one by one. Only the frames classified as showing activity are further processed through stage 2. Output of the system is a time series as discussed in Section 3.1

3.3. Stage 1

The goal of this stage is to filter non-activity frames from the activity frames. Thus, it is modeled as a background subtraction problem. Figure 2 shows the block diagram of background subtraction method.

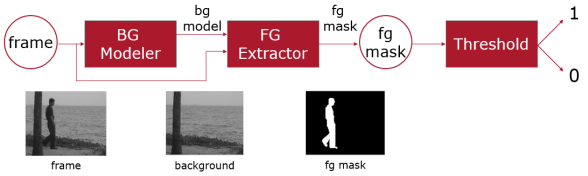

Figure 2: Stage 1 - Background subtraction model

We attempt to model the background as mixture of gaussians [15, 16]. Since, an image usually represents many different surfaces/objects, each surface/object is expected to give rise to a new gaussian. Thus all the pixel values are better represented by a mixture (sum) of gaussians. Notice that this model represents both foreground and background simultaneously. In order to apply this model to background subtraction problem, we associate each pixel with a particular surface and then associate that surface with either foreground or background. The label of each pixel (foreground/background) is determined by the label of corresponding surface.

3.3.1. Background (BG) modeler

Each surface (or uniform object) that comes into the view is represented by a state $k \in 1, 2, 3, ..., K$. Some of these states correspond to background while remaining ones are considered to be foreground. The process $\mathbf{k}$ which generates the states is modeled by parameters set $\{w_1, w_2, ..., w_K\}$ where $w_k = P(k)$. Each of these parameters represents a priori probability of surface k appearing in the image. Further, $\sum_{k=1}^{K} w_k = 1$.

This surface process $\mathbf{k}$ is hidden and is only indirectly observable through pixel value process X. The pixel value process $\mathbf{X}$ is an observable random variable modeled by a gaussian process for given surface k. $\mathbf{X}$ is 1-D in case of gray scale images and 3-D for color images. If $\theta_k = \{\mu_k, \Sigma_k\}$ represent the associated gaussian process then pixel value process $\mathbf{X}$ given k is:

$$f_{\mathbf{X}|k}(X|k, \theta_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} e^{-\frac{1}{2}(X-\mu_k)^T \Sigma_k^{-1} (X-\mu_k)}$$

where μ_k is the mean and Σ_k is the covariance matrix of associated k^{th} gaussian.

We assume these k events are disjoint so $\mathbf{X}$ can be modelled as sum of gaussians.

$$f_{\mathbf{X}}(X|\Phi) = \sum_{k=1}^{K} w_k f_{\mathbf{X}|k}(X|k, \theta_k)$$

where $\Phi = \{w_1, \mu_1, \Sigma_1, ..., w_K, \mu_K, \Sigma_K\}$.

3.3.2. Foreground (FG) modeler

In order to apply the model to background subtraction problem, first step is to determine which of the K states is most likely to give rise to current pixel value $\mathbf{X} = X$. The posterior probability $P(k|X, \Phi)$ is the likelihood that pixel value X was generated by surface k. Using the Bayes's theorem:

$$P(k|X, \Phi) = \frac{P(k) f_{\mathbf{X}|k}(X|k, \Phi)}{f_{\mathbf{X}}(X, \Phi)}$$

The k which maximizes the $P(k|X, \Phi)$ is considered to be the surface associated with X.

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, P(k|X, \Phi)$$

Once X has been associated with a particular surface $\hat{k}$, it needs to be determined whether $\hat{k}$ is a foreground surface or background.

The procedure for demarcation starts with ranking K states by $w_k/|\Sigma_k|$ in decreasing order. This ratio is proportional to height of weighted distribution $w_k f_{\mathbf{X}|k}(X|k, \theta_k)$. A surface k is considered to be background if it occurs more frequently (higher w_k) and does not vary much (low $|\Sigma_k|$). To separate the foreground and background surfaces, an overall prior probability T of anything being in the background is used. The first B of the ranked states whose accumulated probability crosses the threshold T are considered to be background.

$$B = \underset{b}{\operatorname{argmin}}(\sum_{j=1}^{b} w_j > T)$$

3.3.3. Threshold

Output of foreground extractor is a binary mask which indicates whether a pixel belongs to foreground or not. All the foreground pixels in the image can be summed up and their ratio to the the total number of pixels in frame can be compared to a threshold value τ. If the ratio is greater than threshold, then this frame is regarded as activity frame (1); otherwise it is classified as background frame (0).

3.4. Stage 2

The goal of this stage is to verify human trespassing in case of activity reported by the stage 1. State-of-the-art deep learning model Faster-RCNN[17] is used to model this stage. Faster-RCNN predicts the objects (their labels and locations) corresponding to input frame. The given image is labelled with *trespassing activity* if there is at least on person predicted by Faster-RCNN with corresponding confidence score

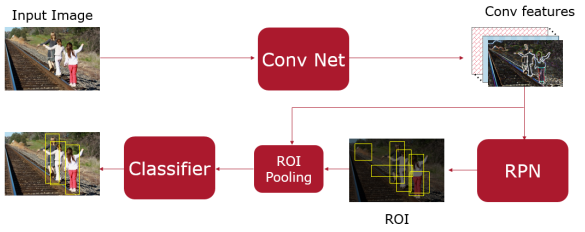

Figure 3: Stage 2 - Faster-RCNN block diagram

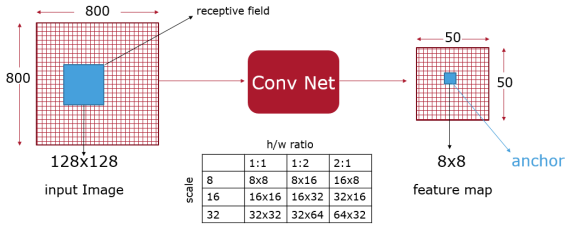

Figure 4: Anchor illustration

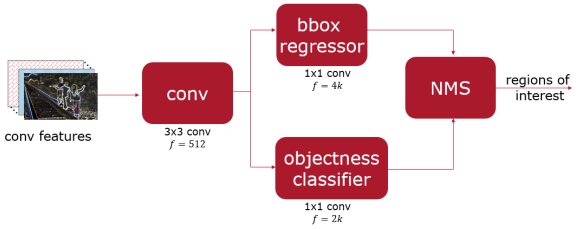

Figure 5: Region Proposal Network architecture

greater than a threshold μ. Figure 3 shows block diagram of Faster-RCNN with detailed explanation in the following subsections.

3.4.1. *Feature extraction (Conv Net)*

First step is to extract convolutional features (also known as feature map) from the input frame/image. These convolutional features will be used by subsequent sub-stages as input. Due to the paramount importance of features generated by this sub-stage, it is also known as backbone of Faster-RCNN. It can be modelled as VGG[18] or Resnet[19]. We use Resnet-50 network with Feature Pyramidal Network (FPN)[20] in our experiments.

3.4.2. *Region Proposal Network (RPN)*

This sub-stage as the name suggests is responsible for proposing regions (rectangles) potentially containing objects (people, cars etc). As seen in Figure 3, this sub-stage takes in the feature map and produces a list of proposals for the given image. Each proposal consists of binary label and proposed bounding box of the region of interest. The label indicates whether the proposal corresponds to an object or background.

a. Anchors

Anchor act as default region proposals. Their idea has been motivated from multi-scale sliding windows. Suppose we use a feature extraction convolutional network such that it converts a 800×800 image to 50×50 feature map (Figure 4). This means every (x, y) location on feature map corresponds to 16×16 patch/window on original image. Similarly, 8×8 window on feature map corresponds to 128×128 window on original image. This 8×8 window on feature map is known as anchor. Faster-RCNN proposes multi-scale, multi-aspect ratio anchors. A total of 3 scales (8, 16, 32 on feature map) with 3 aspect ratios $(1:1, 1:2, 2:1)$ produces 9 anchors on each (x, y) location on feature map. Since, we have 50×50 locations, therefore this setting produces 22,500 anchors in total. However, in practice we use less than that. All the anchors whose regions lie outside feature map (eg. anchors near edges); they don't participate in training the network.

b. Architecture

Figure 5 shows the architecture of RPN sub-network. Input to this network is the features generated by backbone network. These features are passed through a 3×3 "same[1]" convolution layer. Faster-RCNN uses 512 output feature depth for this layer. Output of this layer is fed to the bounding box regressor layer and objectness layer which predicts bounding box locations and objectness score simultaneously. Both of these layers are modeled with 1×1 convolution. Bounding box regressor layer has $4k$ output depth where k is the number of anchors and 4 follows from the fact that each proposal is defined by 4 scalar values. For similar reasons, objectness layer has $2k$ output features. Thus each anchor produces a proposal. All of these proposals are post-processed by Non Maximum Suppression (NMS).

3.4.3. *Non maximum suppression (NMS)*

NMS is responsible for removing the duplicate predictions. Figure 6 illustrates the goal of this process graphically. In order to suppress duplicate proposal predictions with the less confidence, first step is to sort all the proposals in descending order. The first proposal is made the reference proposal and pushed to "keep" list.

[1] (h, w) of input and output feature map remains same by automatic padding

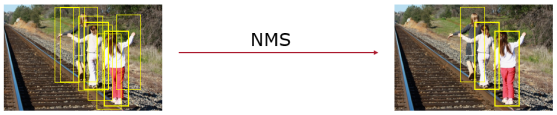

Figure 6: Non Maximum Suppression (NMS). Highly overlapping predictions with lesser confidence are suppressed.

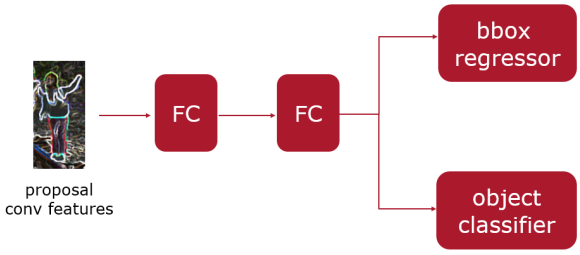

Figure 7: Fast-RCNN classifier

Intersection over Union (IoU) of this reference proposal with all the remaining proposals is computed and the proposals which sufficiently overlap with the reference proposal ($IoU > 0.7$) are discarded. They are considered to be the duplicate of reference proposal. In the next iteration, the first proposal in the list of undecided proposals is made reference proposal and the process of first iteration is repeated. Again this leads to removal of all the proposals considered to be duplicate of reference proposal. The process continues until all the proposals are decided i.e. either kept or discarded. Output of this process is the list of kept proposals.

3.4.4. Fast-RCNN head

Once we have the proposals from RPN, we need to predict the corresponding objects' labels and location. Faster-RCNN uses Fast-RCNN head[21] for this purpose. Fast-RCNN head has two further subcomponents. First one is Region of Interest (RoI) pooling and second one is classifier layer (Figure 3). Input to the Fast-RCNN head will be feature map and list of kept proposals, and output shall be improved bounding box locations of corresponding proposals along with class labels.

Different proposals have different feature map sizes. However, the classifier expects them to be of same size. This process (RoI pooling) is responsible for converting variable sized feature maps into fixed sized. The methodology used by Fast-RCNN in this case is quite simple. Suppose a feature map of size 8×8 has to be converted to 2×2 size. Then, a grid of size 2×2 is placed on top of feature map such that its boundaries align with the feature map. The maximum feature value from each grid cell is copied to corresponding cell in output buffer. This converts a 8×8 feature map to a size of 2×2.

Once RoI pooling has adjusted the size of feature map to fixed dimensions, the feature maps are ready to be fed to classifier. The classifier takes in those features and pass them through two fully connected layers. The output of those two layers is fed to two separate fully connected layers responsible for predicting bounding boxes and object class labels. The bounding boxes and labels so predicted are the final output of Faster-RCNN. Figure 7 illustrates the architecture of classifier.

3.4.5. Training loss

While training the Faster-RCNN, we train two subnetworks: RPN and classifier. Both of these networks have two objectives: label classification and bounding box regression. Following two equations indicate the RPN and classifier loss. First term corresponds to label classification and second term corresponds to bounding box regression.

$$Loss_{RPN} = \frac{1}{N_{cls}} \sum_i L_{cls}(\hat{p}_i, p_i) + \frac{\lambda_r}{N_{reg}} \sum_i p_i L_{reg}(\hat{t}_i, t_i)$$

$$Loss_{cla} = \frac{1}{N_{cls}} \sum_i L_{cls}(\hat{q}_i, q_i) + \frac{\lambda_c}{N_{reg}} \sum_i [q_i > 0] L_{reg}(\hat{u}_i, u_i)$$

where
$\hat{p}_i$ = anchor label prediction
$\hat{t}_i$ = anchor bounding box prediction
$\hat{q}_i$ = RoI label prediction
$\hat{u}_i$ = RoI bounding box prediction
p_i = anchor ground truth label
t_i = anchor ground truth bounding box
q_i = RoI ground truth label
u_i = RoI ground truth bounding box
λ_r = RPN loss balance coef.
λ_c = classifier loss balance coef.
N_{cls} = mini-batch size
N_{reg} = total anchors

Classification loss L_{cls} is the standard log loss and regression loss L_{reg} is the smooth-L_1 loss as defined below.

$$L_{cls}(\hat{y}, y) = -\sum_j y_j log(\hat{y}_j)$$

$$L_{reg}(\hat{b}, b) = \sum_{j=1}^{4} S\,L_1(\hat{b}_j - b_j)$$

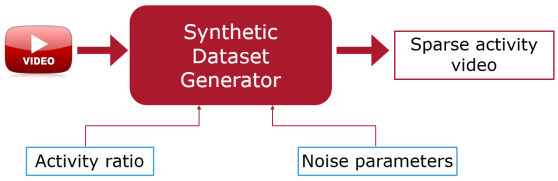

Figure 8: Synthetic dataset generator

$$SL_1(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ x - 0.5, & \text{otherwise} \end{cases}$$

where input parameters to each function carry standard meaning.

Total loss of the network is simply the sum of RPN loss and classifier loss.

$$Loss = Loss_{RPN} + Loss_{cla}$$

Apart from that notice the term p_i in regression term of RPN loss. This makes sure that regression loss is activated only if proposal corresponds to an object, not the background. Thus bounding box predictions corresponding to background proposals do not contribute towards training. Furthermore, the expression $[q_i > 0]$ does a similar job in classifier loss. This again acts as a flag to add regression loss only corresponding to actual objects and not the background.

λ_r and λ_c act as the balancing parameter between label classification and bounding box regression. The authors of Faster-RCNN claim that λ_r is redundant and Faster-RCNN remains insensitive to a large range of λ_r.

4. Dataset

The dataset we used for experimental evaluation is VIRAT 2.0 [22]. Instead of directly using the VIRAT 2.0 dataset, we synthesize more data from it. The reason why we need to do that is original dataset targets activity classification and is therefore enriched with human activity. We on the other hand need sparse activity data. Therefore, we use original VIRAT 2.0 dataset to generate new data that has controlled amount of activity to background ratio. Figure 8(removable) helps understand our synthetic dataset generator.

In order to generate the synthetic data, we follow the following steps:

1. identify background frame
2. make background block
3. write background block(s)
4. write activity block

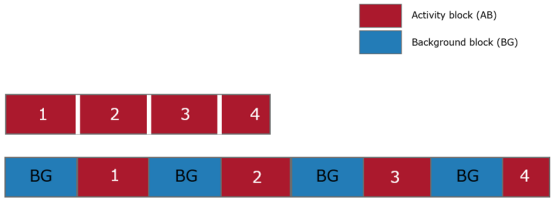

Figure 9: Synthetic video illustration

Table 1: Relationship of number of background blocks and activity ratio

nBG	nAB:nBG	AR
1	1:1	0.5
3	1:3	0.25
5	1:5	0.16

5. repeat (3) and (4)

Step 1 is simple. We manually scroll through a video and identify a frame with no activity (ideally no person). In step 2, we repeat the frame $\Delta \times fps$ times where fps is the frame-per-second of original video and Δ is the target length of background block in seconds. Then we alternatively write background block(s) and activity block in turn. An activity block is simply a section of original video. By default, we use $30s$ long activity blocks. Length of background block is also kept at $30s$ by default. Figure 9 helps understand the concept of activity block and synthetic video.

An important concept associated with this procedure is *Activity Ratio*. It is defined as

$$\text{Activity Ratio} = \text{AR} = \frac{1}{1 + nBG}$$

where nBG = number of background blocks per activity block. Figure 9 has $nBG = 1$. Table 1 (removable) explains the relationship between nBG and AR.

4.0.1. Adding noise

In order to test the robustness of our approach, we also add noise to our synthetic data. Table 2 discuses the parameters that control the level of noise. We use **salt and pepper** noise in our experiments.

In order to add noise, we first select $p\%$ of frames from each background block. Each of the frame is equally likely to be selected. Now we draw an integer r from normal distribution with parameters μ and σ. Now for each selected frame, we select r pixels and add noise to them. Again all pixels are equally likely.

Table 2: Noise parameters

params	description
p	percentage of noisy frames in BG block
μ	avg. no. of noisy pixels in noisy frame
σ	standard deviation of no. of noisy pixels

5. Experiments and discussion

In order to validate our approach, we carry out an extensive and in depth experimental evaluation. We study our proposed approach from three different perspectives:

1. time-accuracy trade-off
2. stage-wise analysis
3. noise analysis

We shall start by discussing time-accuracy trade-off. This allows us to study end-to-end performance of pipeline while trading off computational time. We also do stage-wise analysis where each stage is evaluated independently of other. This helps us understand which stage is acting as bottleneck in terms of performance. Additionally, we also do noise analysis to understand the robustness of system to noise.

5.1. Time-accuracy trade-off experiment

In time-accuracy trade off experiment, we study how the data can be processed in less time by compromising performance. We vary certain control parameters (stage 1 threshold in this case) and observe the time it takes to process the data along with the performance of the complete pipeline. Figure 10 shows time-accuracy trade off for varying activity ratios (AR). The trade off curve depicts f1 score on y-axis and normalized processing time on the x-axis. Next we give the definition of normalized processing time.

$$\text{normalized time} = \frac{\text{total time to process video data}}{\text{length of video data}}$$

It is clear that as the f1 score goes up, the normalized processing time also goes up, indicating the trade off. Further, notice that the trade off curve shifts to the left as AR decreases. This is expected as stage 1 filters more and more frames with increasing threshold and thus lesser frames are processed by stage 2.The synthetic dataset parameters using in this experiment are shown in table 3.

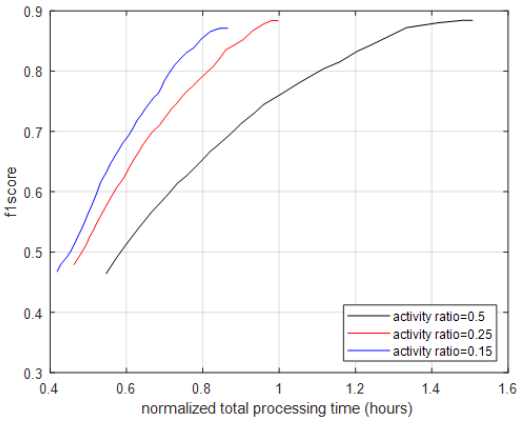

Figure 10: Time-accuracy trade off for varying AR

Table 3: Synthetic data parameters for time-accuracy trade off

parameter	value
p	1%
μ	0.50%
σ	0.20%

5.2. Stage-wise analysis experiments

In this part, we analyse both stage 1 and 2 independently. We use AUC (Area Under the Curve) as the evaluation metric since it is not affected by class imbalance. We naturally have class imbalance since very few trespassing frames exist as compared to other frames. Table 4 shows that stage 1 takes only 30 ms where as stage 2 takes around 500 ms on 1080×960 sized frame. This shows that stage 1 is approx. 16.7 times faster than stage 2 wich resonates with our goal stated in Section 1.2.

stage 2 methodoloy-if space permits

5.3. Noise analysis experiments

Third and final part of our experimental evaluation is noise analysis. In this part, we evaluate the robustness of stage 1 against noise. In terms of noise, we have 3 parameters: p, μ, and σ. Description of these parameters has been discussed in table 2. To study the influence

Table 4: Stage 1 AUC and time analysis

stage	AUC	mean processing time (ms)
stage 1	0.94	30
stage 2	0.81	500

Table 5: Noise analysis - AUC for varying p and μ

p	AUC	μ	AUC
2%	0.93	0.5%	0.93
4%	0.87	0.7%	0.92
6%	0.83	0.9%	0.91
8%	0.78	1.1%	0.88

of one parameter, we vary it while keeping the others constant.

Table 5 shows AUC for stage 1 varying p and μ. We use the noise parameters ($AR = 0.5$, $\mu = 0.5\%$, $\sigma = 0.2\%$) while varying p and ($AR = 0.5$, $p = 2\%$, $\sigma = 0.2\%$) while varying μ. For both p and μ, it is clear that increase in noise level decreases the performance of stage 1.

6. Conclusion and future work

In this work, we propose a flexible trespassing detection system that can trade off speed and performance. Although initially envisioned for railroad security, the proposed approach has potential applications in video surveillance domains characterized by a sparsity in activity. The system by design consists of two stages where the first stage is responsible for efficiently removing the background frames from the activity frames. The second stage is responsible for differentiating between human trespassing activity and any other unknown activity. Our proposed pipeline is composed of off-the-shelf components, allowing researchers to plug in other algorithms relevant to stage 1 and stage 2. The effectiveness of the approach has been demonstrated on a public domain surveillance dataset.

For the future, one key direction is to build a trespassing prediction system that uses the output of this detection system to predict trespassing events in near future. Another direction is towards improving the accuracy of detection system. We note that the current performance is limited by the performance of stage 2. Currently stage 2 does not use any temporal information i.e. each frame is treated independently and is not conditioned on the previous frames (history). We believe that utilizing the temporal information can significantly improve the performance specially for challenging cases of occlusion and background.

References

[1] J. Caird, Human factors analysis of highway-railway grade crossing accidents in canada (2002).

[2] E. D. Sussman, T. G. Raslear, Railroad human factors, Reviews of human factors and ergonomics 3 (2007) 148–189.

[3] R. D. Austin, J. L. Carson, An alternative accident prediction model for highway-rail interfaces, Accident Analysis & Prevention 34 (2002) 31–42.

[4] B. Tustin, H. Richards, H. McGee, R. Patterson, Railroad-highway grade crossing handbook, Technical Report, United States. Federal Highway Administration, 1986.

[5] S. G. Chadwick, N. Zhou, M. R. Saat, Highway-rail grade crossing safety challenges for shared operations of high-speed passenger and heavy freight rail in the us, Safety Science 68 (2014) 128–137.

[6] F. Saccomanno, L. Fu, C. Ren, L. Miranda, Identifying highway-railway grade crossing black spots: phase 1, Technical Report, 2003.

[7] M. Chaudhary, A. Hellman, T. Ngamdung, et al., Railroad right-of-way incident analysis research., Technical Report, United States. Federal Railroad Administration, 2011.

[8] J. Glover, The Level Crossing Conundrum, Technical Report, Rail Professional (November), 2009.

[9] S. Hall, Reducing risk at automatically operated level crossings on public roads (2007).

[10] S. M. Horton, et al., Use of traffic channelization devices at highway-rail grade crossings, Technical Report, United States. Federal Railroad Administration. Office of Research and ..., 2012.

[11] M. Shah, O. Javed, K. Shafique, Automated visual surveillance in realistic scenarios, IEEE MultiMedia 14 (2007) 30–39.

[12] H. Salmane, L. Khoudour, Y. Ruichek, A video-analysis-based railway–road safety system for detecting hazard situations at level crossings, IEEE transactions on intelligent transportation systems 16 (2015) 596–609.

[13] Z. Zhang, C. Trivedi, X. Liu, Automated detection of grade-crossing-trespassing near misses based on computer vision analysis of surveillance video data, Safety science 110 (2018) 276–285.

[14] R. Benenson, M. Omran, J. Hosang, B. Schiele, Ten years of pedestrian detection, what have we learned?, in: European Conference on Computer Vision, Springer, pp. 613–627.

[15] C. Stauffer, W. E. L. Grimson, Adaptive background mixture models for real-time tracking, in: Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), volume 2, IEEE, pp. 246–252.

[16] P. W. Power, J. A. Schoonees, Understanding background mixture models for foreground segmentation, in: Proceedings image and vision computing New Zealand, volume 2002.

[17] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, pp. 91–99.

[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[20] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125.

[21] R. Girshick, Fast r-cnn, in: The IEEE International Conference on Computer Vision (ICCV).

[22] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, et al., A large-scale benchmark dataset for event recognition in surveillance video, in: CVPR 2011, IEEE, pp. 3153–3160.