TRAFFIC ANOMALY DETECTION

Bernat G. Škrabec
Student, M.Sc. DSE
Eurecom, Sophia Antipolis
Bernat.Gene-Skrabec@eurecom.fr

Maxime Michel
Student M.Sc DSE
Eurecom, Sophia Antipolis
Maxime.Michel@eurecom.fr

Matteo Nottaris
Student M.Sc DSE
Eurecom, Sophia Antipolis
Matteo.Nottaris@eurecom.fr

July 2, 2021

Supervisors: Prof. Jean-Luc Dugelay, Mohamed Jamel Eddine

ABSTRACT

With the rise of modern AI tools, intelligent video analysis can be leveraged to make cities smarter and safer. In particular, transportation is one of the segments where building intelligent systems, using data captured by cameras and other sensors, could provide enormous benefits. Within the context of the 5th Edition of NVIDIA's AI City Challenge, we explore methods for anomaly detection in traffic videos. The aim is to autonomously detect anomalous behaviour ranging from stalled vehicles in the highway to multi-vehicle accidents, in order to quickly rise alarms that could cut down the reaction time needed for response teams to deploy. We present our insights, gathered during the research for such a system; techniques, challenges and caveats, and we describe a simple solution which, while flawed, can help understand the intricacies of the problem and serve as a base for more sophisticated systems.

1 Introduction

Nowadays video surveillance systems are established in a large majority of public areas, including roads and highways. Their goal is to provide security and a record for insurance or police investigations.

The increasing number of vehicles due to population growth makes traffic monitoring and control very difficult for a human agent. Fortunately, as research points out, transportation is one of the largest segments that can benefit from useful insights derived from data captured by sensors (e.g. video surveillance). The improvement of computer vision techniques allows the fast development of automatic detection systems for incident, the goal being to inform a surveillance agent of any possible danger. With this information the agent could trigger an alert in a very short time without constant human monitoring of the scene.

In our research, we are interested in the detection of anomalies on highways through different computer vision and machine learning techniques, using low quality videos from the large number of cameras established on the side of the roads.

2 Data Description

The used data comes from AIC21 Track 4 Dataset [1]. It is composed of 100 training and 150 test videos of 15 minutes each. All those videos are from traffic cameras, with 410p resolution feeds at 30 fps. The anomalies are of two types, crashed cars and stalled vehicles. The data is strongly imbalanced, in the 100 training (labelled) videos only 18 contain anomalies including only 2 accidents/crashes, the rest being stalled vehicles.

3 State of the art

A large part of the state of the art for this particular goal are product of the work on the 4 first editions of the Nvidia's challenge. Most of the state of the art methods for traffic anomalies are based on a analysis of the vehicle trajectory. The process aims at comparing the behaviour of the vehicle with the other vehicles of the highway. In order to do that there are two main ways. The first one proposed in [2] is analysing the surrounding environment, checking if the mean speed changes in a particular region of the video, or checking if a group of vehicles changes their trajectory suddenly. The second one, proposed in [3] is about focusing on each vehicle and checking if its own behaviour differs from what it is suppose to do.

The 4th edition of the Nvidia challenge, summarized in [4], brought forward that the first approach gave the best results with a F1 score of 0.96. The second best approach was a fast unsupervised system using K-means clustering. While we explored and use some of the ideas presented by previous participants to this challenge, we decided against reproducing (i.e. copying) any particular approach and instead attempted to develop our own approach based on our understanding of the problem. While this decision may hinder our chances of delivering a successful solution, we are forced to achieve a deeper understanding of the problem, which results in a better learning experience; the actual objective of this project.

4 Limitations

The first source of difficulties is the data itself. Indeed a large part of the videos are of a very low quality and that creates issues, mostly of 3 types :

- Too much depth : For a lot of the videos a part of the captured highway is far away and therefore the cars are small points that are very difficult to detect. In some cases the anomaly happens too far to be correctly distinguished, even by a person.
- Bad resolution : Several videos are taken during the night or in bad weather. This, added to the original quality of the video, gives very blurry images for which even a human being could have difficulties discerning individual vehicles and their position.
- Defaults : A part of the videos includes defaults like a vibrating camera or a lot of freezing. Those defaults make the analysing process very complicated.

5 Our Method

5.1 Motivations

Our goal was to find an efficient way to detect anomalies, but also a very transparent one: a method that can be understood and customised in every of its aspects. That is what led us to choose a

three-steps clear process. Moreover, our resources are limited and our team didn't have access to any powerful hardware as GPUs, therefore, our method needed to be light so it could run on our laptops.

5.2 Description

Our key idea is to model the ordinary world with a predictive model and then compare the predicted behaviour during a time window with the real behaviour to determine if an abnormal event occurs. Our process is divided in three parts, described on the figure below: 1. The next figure, 2, shows the detailed steps of the whole pipleine, which are described in detail in the following sections.

5.2.1 Vehicle Detection and Tracking

The first step of the process consists in detecting each vehicle appearing on the frame and tracking them. The goal is to gather data about the trajectories of every moving object that corresponds to a vehicle, but ignore other irrelevant movements. In order to do that we used Lucas-Kanade's Optical Flow algorithm. In particular, we used OpenCV's implementation [5]. The Lucas-Kanade optical flow algorithm is a simple technique which can provide an estimate of the movement in successive images of a scene. It allows to associate a movement vector (u, v) to every vehicle in the scene, obtained by comparing the two consecutive images, and to each moving detected pixel assign a point and follow it. By setting certain thresholds about the length and direction of this vectors, we can filter out most background movement (trees, clouds, etc.) and remain only with the actual vehicles. This tuning process has been done experimentally.

5.2.2 Predictive Process

The second step is the working horse of the system, in which we try to predict the next positions of a track, given the previous positions observed so far. There are many methods which could solve this task, but we decided to go for an LSTM, Long Short-Term Memory [6]. LSTM's are a type of RNN (Recurrent Neural Network) very well suited for sequence modelling. The main idea behind an LSTM is that it keeps track of a hidden state over arbitrary long periods of time, so it has a better capacity to model the time dimension. In our particular case, this means that it can use more than a few of the previous points to know what the next one should be.

Given the lack of data available to us, and the absolute disparity between different scenarios, we were not able to train a general model that could work in all scenarios. Instead, and we understand that this is one of the major caveats of our model, we use the first 2-3 minutes of the chosen videos as a *nominal* training data, in which we know that there are no anomalies. Then, the model is used to predict the next 10-12 minutes of unseen video.

The justification we can give for this *tuning* step is the following: We can imagine that when a model is deployed in a given environment, a human can spend some time actively surveying the environment to ensure that no anomalies are present in the data. This will tune the model to the particular environment. Furthermore, any missed anomalies will be recorded and probably eventually reported, so a sort of delayed online learning could be use to further improve the capacity of the model on the given environment, the more time it spends deployed.

All that being said, this makes us deviate from the challenge's rules, but we could not find a way to overcome this problem.

Pipelines

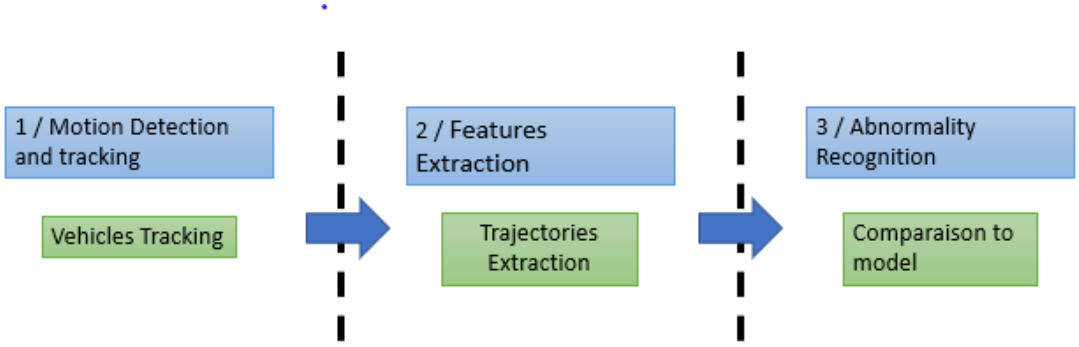

Figure 1: Theoretical Pipeline

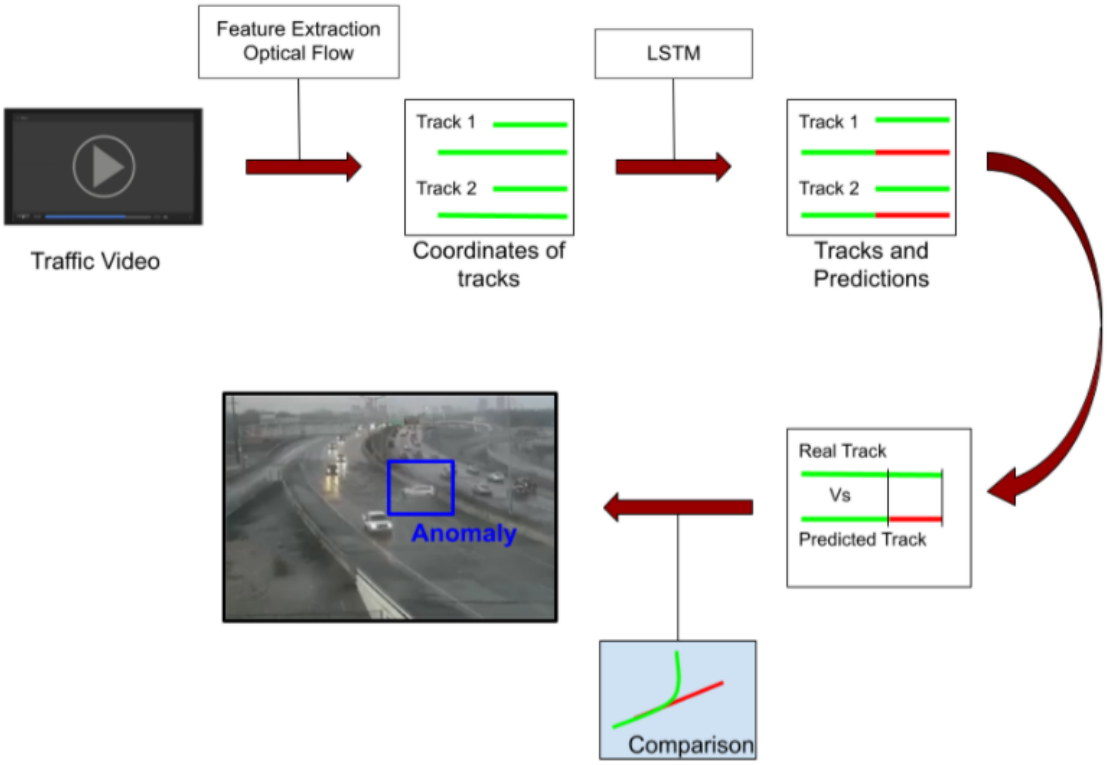

Figure 2: Actual Pipeline

5.2.3 Model architecture

To construct our model, we used the Keras library [7], which is a deep learning API written in Python very useful for fast experimentation. The model architecture was built as follows. We defined a useful input and output dimension, 20 points for input and 10 points for output.

Then, we built the simplest model that would be able to use this data format. We trained for some epochs and recorded the lowest achieved loss. After that, we incrementally added complexity to the model, while monitoring the validation loss, that is, the performance of the model on unseen data.

The final architecture is described in the following figure 3. As said, we have an input dimension of 20 consecutive points, X, (which are 2D image coordinates) and the next 10 points, which are the target, y. The input tensor is passed through an LSTM block which outputs another tensor with a higher dimension. This feature tensor/vector is then passed through a couple RELu activated, fully connected layers, plus a final fully connected layer which outputs the target dimension (the next 10 points) and has no activation as our output is more or less unbounded. Both the target ground truth and the prediction tensors are flattened for convenience, but this does not have any effect, as the semantics of each item are preserved.

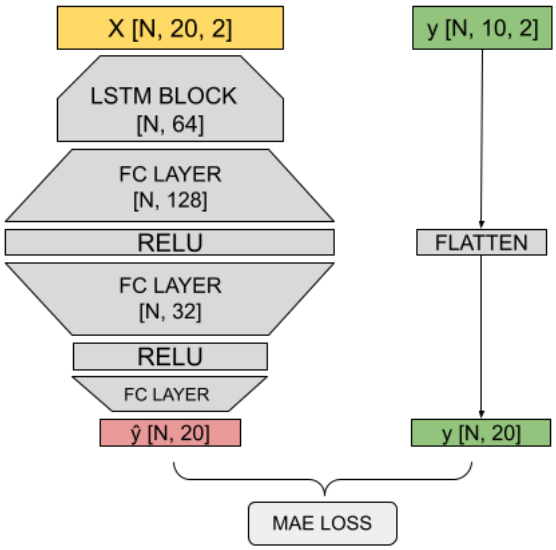

Figure 3: Model architecture

5.2.4 Abnormality Recognition

The last step consists (finally) in detecting the anomaly itself. In order to do that, our system compares the predicted tracks from the LSTM to the actual one. At every single moment and for each tracked point this comparison is made for the last 10 coordinates (previously predicted). The detector focuses on two aspects, speed and trajectory. For both of those characteristics the difference between real and predicted is compared to 2 pre-defined thresholds. If the predicted value is too different from the actual one then an alarm is triggered and the abnormality indicated immediately on the video.

Note : The thresholds have been defined empirically for this version of the project, but are not unique to the environment, i.e. we used the same threshold on all our results.

5.3 Experiment

The data set includes 18 anomalies video and 82 normal ones. Because of the way our pipeline worked, we ended up testing our system only on a handful of the videos. On 18 videos including an anomaly 11 were discarded because of quality issues or because they were incompatible with our method (The anomaly started in the first frame, and thus we had no *nominal* scenario to use for training), 4 were detecting the problem, of which 2 had some inaccuracy and false positives (anomalies outside the actual location of the accident) and finally in 3 videos the system didn't

work. The current version is much more efficient on stalled cars than actual accident with trajectory issues, but this statement holds little power as we only had 2 accident videos, and both presented serious defaults and video-corruption.

In any case, almost all of the the errors that the system made were due to the lack of quality in the videos, indeed the approach we use is unable to counter video freezing, bad luminosity or poor image quality (where the vehicle are difficult to distinguish). In the example of failure given below the error is due to the blurry aspect of the image that make it difficult to predict a correct trajectory and therefore make the whole process of comparison work.

Figure 4: Successes Screenshots

Figure 5: Failure Screenshot

For the videos without any anomalies, the ones we tested with a good quality, our system worked well, as no anomalies were detected, except for three of them where anomalies were detected where there were none.

While this is a very rough calculation, given that, first, we did not test it on all of the videos, and second, we only tested in the training set, we did the following: First, we consider the F1-score as if only the videos that we decided to treat existed, then, we consider the global F1-score, in which we assume that our system misses all the anomalies of the untreated videos and has a 0.5 chance of going off in the videos without an anomaly, as an approximation of it's capabilities. Also, since for one video we can both detect the anomaly and detect a false anomaly, the definition of f1-score is as follows: The precision is defined as the times an anomaly detection was correct over the times an anomaly was raised. The recall is the amount of anomalies detected over the total of existing ones.

$$f1 - score = 2 \times \frac{p \times r}{p + r}$$
$$p = precision$$
$$r = recall$$

The next table summarises our results and gives an approximate score to our method.

	Detected correct	Missed	Detected false	No false	Discarded (Bad quality)
Videos with anomalies	4	3	2	2	11
Videos without anomalies	0	0	3	12	67

Table 1: Summary of the results. Please note that for any given video, we can both detect an a correct anomaly and a false one, which is why the rows don't sum to the total amount of videos of each class.

	Missed anomaly	Detected anomaly
Was not anomaly	Undefined	5
True anomaly	3	4
F1-score Treated	0.5	

Table 2: Summary considering only the treated videos

	Missed anomaly	Detected anomaly
Was not anomaly	Undefined	38
True anomaly	14	4
F1-score Global	0.133	

Table 3: Summary considering all of the videos. We assume that our system misses all the anomalies of the untreated videos and has a 0.5 chance of going off in the videos without an anomaly. Please note that this is quite arbitrary and meant only as an approximation, as we did not test our system in said videos.

Our code for the pipeline and the experiments can be found in our GitHub repository [8].

6 Improvement Ideas

6.1 Countering Freezing

The most recurrent reason because of which the videos couldn't be treated was freezing. Indeed some videos were freezing repeatedly around the moment of the anomaly creating a loss of information and difficulties to track the vehicles. The few following solutions where thought of but not implemented :

- Vehicle Re identification
- Video editing to avoid the freezing

6.2 1 track per vehicle

The actual system doesn't care if the same vehicle is tracked multiple times. Indeed the tracking function sometimes track independently several point on the cars. In order to smooth and lighten the process it could be efficient to join those multiple tracking point and have only one per vehicle.

6.3 Use of an ensemble method

An ensemble approach consists in combining multiple methods, it could mitigate the fragile requirement of our current approach (Need of an ideal environment ..). For example the use of an image-based model for vehicle re-identification with in parallel an object detection algorithm to detect people or fire. Ultimately, this approach comes down to backing up our own system with new functions.

7 Conclusion

Thanks to the Nvidia data and subject proposition we have delved into a very challenging problem. Challenging because of the difficult task but also because of the data quality and quantity. During this challenge the team have researched the state of the art and studied multiple tools like Optical Flow, Object Detection or deep learning sequence modelling. The solution produced by our team is as we wanted it, very transparent and simple, in which all the parts are clear, understandable and manipulable, but it does require an ideal scenario. Given our experience in the domain it's the best we could do.

The whole experience has been very formative, it led us to learn by trying and even if the result is far from perfect it gave use some very useful knowledge.

References

[1] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E. Lopez, Anuj Sharma, Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. The 5th ai city challenge, 2021.

[2] Abdulhamit Subasi Nejdet Dogru. Traffic accident detection by using machine learning techniques, 2016.

[3] M. Do M. Tran Khac-Tuan Nguyen, Dat-Thanh Dinh. Anomaly detection in traffic surveillance videos with gan-based future frame prediction, 2020.

[4] Milind Naphade, Shuo Wang, David Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge, 2020.

[5] OpenCV. Optical flow. `hhttps://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html`, 2021.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[7] François Chollet et al. Keras. `https://keras.io`, 2015.

[8] The Authors. Traffic anomaly, project code. `https://github.com/bernatGene/trafficanomaly`, 2021.