11-Foot-8 Crash Detection

Sarah Habib, Tyler Feldman, Rodrigo de Albuquerque, Shaan Gondalia
Duke University

1 Background

Detecting anomalous behavior using image and video processing is a large area of interest across many disciplines. Traditionally, we have relied on human observation to identify abnormal behavior. Anomaly detection via image and video processing can automate this process, reducing the human effort and time required to react to potential issues. A particularly relevant application of these techniques is real-time traffic accident detection. Accurate crash detection can increase the response speed of incident management, which reduces potential injuries and subsequent traffic congestion. Recent developments in traffic infrastructure have resulted in the widespread installation of stationary cameras along streets [1]. Vision-based crash detection can use these feeds to automatically identify crash scenes.

The 11-foot-8 bridge, located in Durham, NC and affectionately named the “Can-Opener,” is notorious for its low clearance that causes large trucks to crash into the bottom of the bridge. Cameras have been set up to record these crashes and many of these videos have been posted to a dedicated website and YouTube channel[2]. The maintainer of this website must manually scrub through the footage to determine if and when a crash has occurred. Automated crash detection through real-time analysis can archive relevant footage, greatly reducing the workload of obtaining and documenting crash footage.

Figure 1: 11 foot 8 bridge

Most modern vision-based crash detection models detect vehicle-to-vehicle accidents, relying on the vehicles to stay in frame for an extended period, allowing the detection of crash features [1]. The architecture of the 11-foot-8 bridge results in the obstruction of the crashed vehicles immediately after collision, causing issues in existing crash detection models. Instead, we propose an alternate model that uses a three-pronged approach to identify crashes. We also provide the code for our system in an open-source repository [3].

2 Methods

The proposed model is a multi-stage architecture that first uses three techniques to analyze relevant features in the footage: structural similarity, template matching, and audio processing. We chose to include several image processing techniques in this first step to increase the robustness of the model. These three modules each report a time-series of crash likelihoods, which are then fed into a neural network that predicts if and when a crash has happened in the footage. The structure of the model is shown in Figure 3.

2.1 Structural Similarity across Crash Bar

The Structural Similarity (SSIM) Index is a metric originally developed for compression that measures the similarity between two given images [4]. Applying the SSIM index to the video gives a temporal measure of stability across the crash bar. Crash events are expected to be periods of lower stability.

$$\mathrm{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{1}$$

Applying SSIM to entire frames of the video raises several issues. Calculating SSIM for a 1080p video on consumer hardware iterates slower than the real-time frame-rate. Also, frame-wise similarity is generally low between entire images in traffic videos, as moving vehicles and pedestrians are present. To resolve these issues, SSIM is only calculated across the "crash bar", a yellow beam with the same height as the 11-foot-8 bridge. Vehicles that crash into the bridge also hit this crash bar, resulting in vibrations and debris that are captured by SSIM.

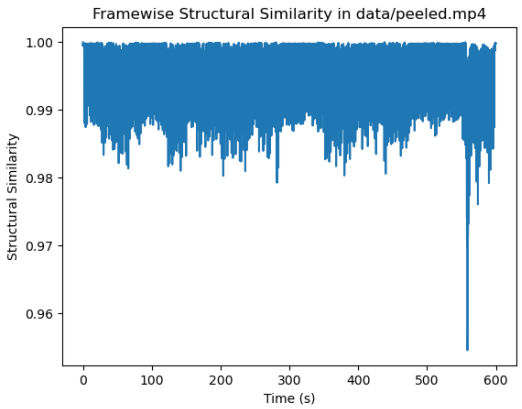

Figure 2: SSIM index comparing each frame its previous. Clear spike during crash event.

The location of the crash bar in the footage is isolated using color masking and contour detection. A binary mask of all yellow objects is created by filtering a certain range of hue values in the HSV color space of the first frame of the footage.

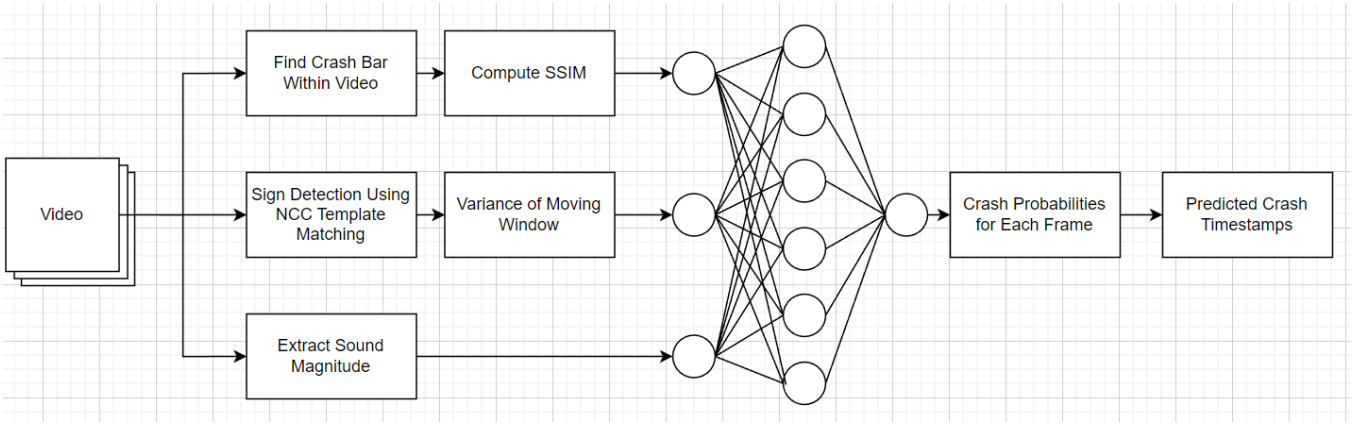

Figure 3: Processing and crash detection pipeline

We find the rectangular contours of the resulting mask using a border following algorithm [3]. We then pick the contour that best matches the expected shape of the crash bar, resulting in a mask that contains only the pixels that contain the crash bar.

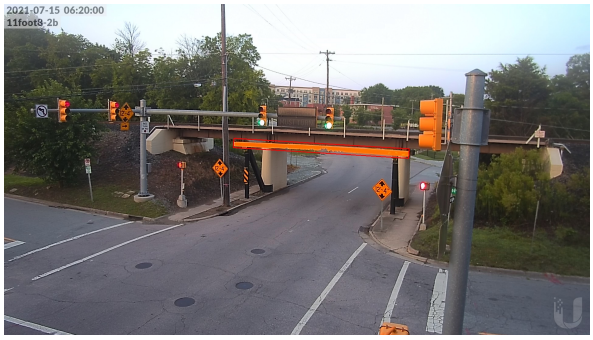

Figure 4: Image with segmented crash bar

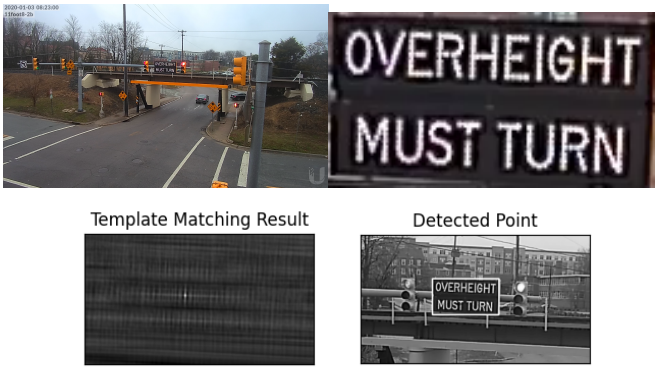

Figure 5: Original Image (top left). Template (top right). Template matching results (bottom)

2.2 Normalized Cross Correlation Template Matching

A sensor setup detects and warns tall vehicles approaching directly from the front of the bridge. In the case that there is a tall vehicle, the sign above the crash bar lights up, displaying the words "OVERHEIGHT MUST TURN". The sign stays on for some time, then starts flashing. We chose to detect when this sign is on because this is a reasonable indicator of when crashes could occur.

We use template matching to detect when the sign is on. First, we manually extracted a template of the sign turned on. We can then slide the template along the image to compute the maximum correlation. This allows detection of the sign no matter where it is in the image, which is important because the camera tends to move in certain circumstances and has slightly different orientations in different footage.

For performance, we use template matching on a selected region of the image rather than the entire image. We additionally compute this for every 5 frames to save computational power.

Because this video is taken outside, there are a variety of lighting conditions and weather to account for. This is why we used normalized cross correlation (NCC), which is resistant to image intensity variations [2]. The equation for NCC is given by equation (2).

$$\mathrm{NCC} = \frac{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]^2 \sum_{x,y}\left[(x-u,y-v) - \bar{t}^2\right]^{0.5}} \quad (2)$$

The sign being on is not a sufficient indicator alone that a crash will happen at that given instance. It only indicates the possibility of a crash around the time the sign is activated, and that the crash may happen when the sign is fully turned on, flashing, or after it has turned off. Because of this, we used an additional processing step where we computed the variance of the cross correlation results over a moving window. By using a large enough window, there is a high variance around both when the sign has just turned on and is flashing, as shown in Figure 6. We then use this moving variance as one of the features fed into our neural network.

2.3 Audio Processing

Most crashes result in loud sounds produced by vehicles colliding with the crash bar. This audio data augments the previous two image processing techniques and increases our model's robustness for footage with poor visibility conditions. We calculate the maximum amplitude across all audio channels per video frame. We normalize this between 0 and 1 to achieve a signal as shown in Figure 7.

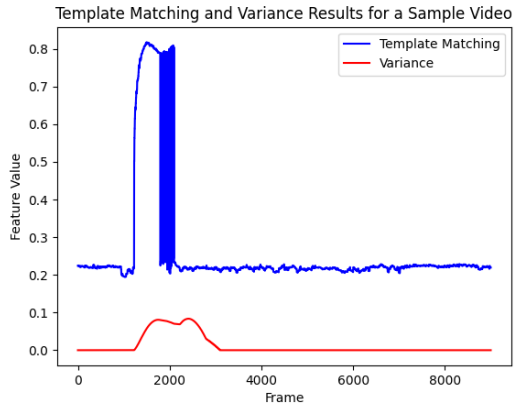

Figure 6: Template matching NCC and Moving Window Variance

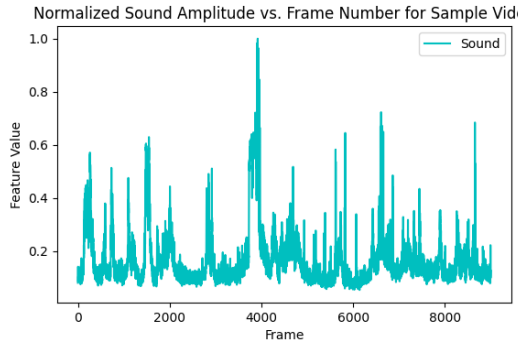

Figure 7: Normalized sound magnitude vs. frame number. Clear spike present at moment of crash around frame 4000.

2.4 Crash Prediction With a Neural Network

After extracting the above three features, we use a neural network to predict whether a crash occurs for a given frame. Specifically, we use a multi-layer perceptron with two hidden layers each with 100 neurons and ReLU activations.

To create a training dataset, we labeled video frames during and neighboring the crash instance as positive examples. This labeled data was used for optimization of the neural network through gradient descent.

The model predicts the probability of a crash in each frame of a given video. When the probability is above a selected threshold, the corresponding frame is classified as a crash. We then sort these predictions by confidence and iteratively prune neighboring predicted frames until we are left with distinct predictions.

3 Results

3.1 Crash Bar Detection

The model accurately identified the crash bar in 19 out of 23 videos, or a success rate of 82%. This portion of the model is highly sensitive to lighting conditions that introduce gaps in the color mask, such as shade and nighttime. In these cases, the resulting contours split the bar into distinct objects that are classified incorrectly.

Figure 8: Contouring results in ideal conditions (left) and shady conditions (right)

3.2 Template Matching Results

To look at the effectiveness of the template matching, we collected and labeled data from our training samples, indicating when the sign was off or on. Below are the normalized cross correlation values for the negative and positive examples plotted as a kernel density estimate. The density for the positive examples is clustered at a much higher correlation value, although there are some positive examples that yield low cross correlation values. Based on this, we generated an ROC curve shown in Figure 10 with an AUC of 0.866, which shows that the sign can be detected 80% of the time with zero false positives.

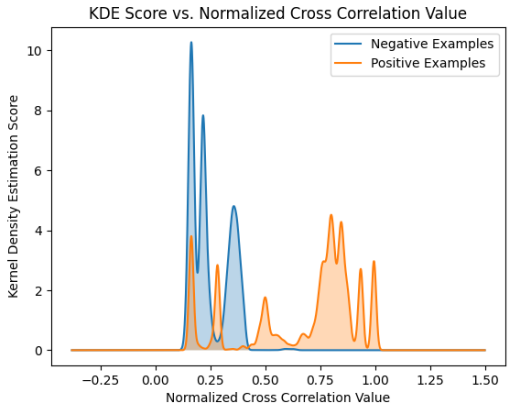

Figure 9: KDE for NCC values

3.3 Crash Detection Results

To determine if we have made a correct prediction, we match the closest predicted crash to each true crash. If the predicted crashes are within a selected threshold of $t_{thresh} = 3$ seconds, the prediction is classified as correct. We can calculate true positives, false positives, and false negatives to compute the precision, recall, and f1 score of the model.

$$precision = \frac{tp}{tp + fp} \quad recall = \frac{tp}{tp + fn} \quad f1 = \frac{2 * p * r}{p + r}$$

To compute the results for our overall crash detection system, we ran cross validation with random train and testing splits for 50 iterations. The average precision and recall are 0.82 and 0.93 respectively.

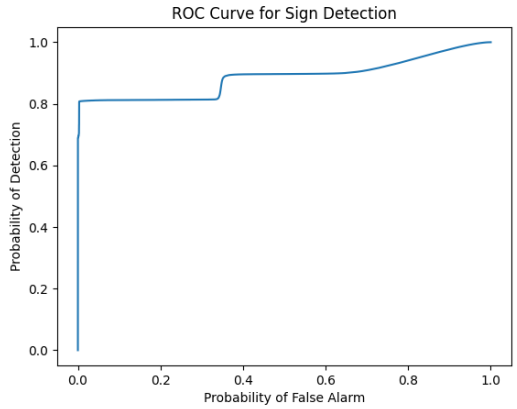

Figure 10: ROC for sign detection

We additionally ran experiments varying the decision threshold. We ran cross validation with random train and testing splits for 20 iterations for decision thresholds ranging from 0 to 1. The precision, recall, and f1 scores for these varying thresholds are shown in Figure 11.

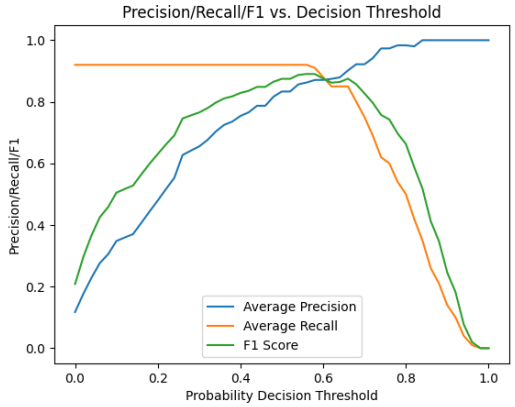

Figure 11: Precision and recall for different decision thresholds

4 Discussion

4.1 Evaluation of Model

For this model to be meaningful to the maintainer of the 11-Foot-8 website, it needs to accurately predict all crashes in the given footage. False positives can easily be filtered out by the maintainer as the model greatly reduces the number of timestamps that need to be checked. As a result, our priority was to maximize recall when building and evaluating our model. Our results show that we can achieve a decent trade-off between recall and precision. We can achieve around 92% recall while having around 86% precision. We would prefer our recall to be closer to 100%, but that would likely require more robust image processing techniques.

The model processes videos 10x faster than the length of the video, which indicates it could be used for both real time crash detection and post-processing.

4.2 Shortcomings

The crash bar detection, SSIM, and sign detection methods all struggle when there are significant variations in lighting and weather. For example, one sample video had snow in front of the sign which made detection more difficult. When the sign was turned on at night, the lights became blurred because of lens flare as shown in Figure 12. Crash bar detection can also be challenging when significant amounts of the paint is chipped away or there are strong shadows. The crash bar SSIM can give poor results when wind causes the camera to move during the video. However, in these cases, the audio data compensated for sub-optimal image processing results. When these other two methods may produce worse results, the audio can still be a useful predictor for crashes.

Figure 12: Lens flare of sign at nighttime (left), and snow on sign plus shadows on the crash bar (right)

4.3 Improvements

To improve our crash detection model, we would need to make our methods more robust to changes in lighting and weather. To achieve this, we would like to try pre-processing techniques such as color correction, image sharpening, and video stabilization before extracting our image processing features.

Collecting a larger and more representative dataset would allow us to adopt a deep learning framework for both audio and video processing like in [1]. Given the dataset provided, we were hesitant to use this approach since it was limited and unbalanced. Given more time, we could scrub or request more crash data and try training a convolutional neural network for image feature extraction and some form of recurrent neural network for audio feature extraction, and then use these features for crash prediction.

To further improve the speed of the system, we could split up the feature extraction into parallel jobs using multiprocessing.

We would additionally like to build a user-friendly interface for processing videos, as well as implement real-time crash monitoring. We currently have functionality to process videos on the command line interface.

Notes

1. https://www.ncsl.org/research/transportation/states-increase-use-of-traffic-cameras-to-counter-surge-in-unsafe-driving-magazine2021.aspx
2. http://11foot8.com
3. https://gitlab.oit.duke.edu/tjf40/11ft8-crash-detection

References

[1] Jae Gyeong Choi, Chan Woo Kong, Gyeongho Kim, and Sunghoon Lim. Car crash detection using ensemble deep learning and multi-

modal data from dashboard cameras. *Expert Systems with Applications*, 183:115400, 2021.

[2] Nazanin Sadat Hashemi, Roya Babaie Aghdam, Atieh Sadat Bayat Ghiasi, and Parastoo Fatemi. Template matching advances and applications in image analysis. *CoRR*, abs/1610.07231, 2016.

[3] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 29(3):396, 1985.

[4] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.