Abstract

Motor accidents can be devastating and cause loss of lives and fatalities. The statistics of the Ministry of Road and Transport show that 1130 accidents and 422 deaths occur every day. Research says that a high proportion of death in road accidents is due to lack of timely medical help. This is a proof-of-concept paper, aiming to display the strength of a framework to detect crashes using CCTV footage. The video input is pre-processed using machine learning techniques Mask RCNN and Deep SORT and a rule-based framework is prepared for detection of accident.

Problem Statement

A major proportion of lives involved in such accidents die or suffer severe fatalities or permanent injuries due to lack of timely help. Damage caused can be reduced significantly if medical help like an ambulance is received within the golden period. However, crashes on lonely roads often go unnoticed and in certain circumstances, even with people around, nobody calls for help immediately. Automating the process of detecting an accident will reduce the time in which the injured people receive medical attention, helping save lives.

Objectives

1. To detect entities in each frame, and their class.
2. To get referenced IDs for each entity and track the entity through successive frames.
3. To determine the speed of each vehicle in each frame.
4. Construction of a rule-based framework which detects accidents.

Literature Review

(Shah et al., 2018) talks about the faster R-CNN model and a dataset of the videos which are extracted for the use case of accident detection. (Ghosh et al., 2019a) discusses the idea to detect an accident taking the live input of video from CCTV cameras. Further, they deploy a deep learning convolution neural network model to categorize video frames into accident-related and non-accident-related categories. (Chakraborty et al., 2018) talks about modern deep learning based on YOLOv3 classifiers that are used to identify car trajectories, and Simple Online Realtime monitoring (SORT) is employed for vehicle monitoring. (Ijjina et al., 2019) suggests a framework that makes use of Mask R-CNN for precise object identification, which is followed by a productive centroid-based object tracking algorithm for security video. It provides a robust method to achieve a High Detection Rate and Low False Alarm Rate. (Chang et al., 2019) undertook the study such that accident detection data is uploaded to the cloud-based database server for self-collision car accident recognition and an associated emergency notice is sent out when a head-on or single-vehicle collision is identified. The trial findings indicate that the precision of traffic accident detection is significantly improved. (Ojha et al., 2021) has a state-of-the-art Mask R-CNN method that is implemented in this article for car identification using instance-wise segmentation, which concurrently generates bounding boxes and object masks. This method uses transfer learning. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world.

Data

Description

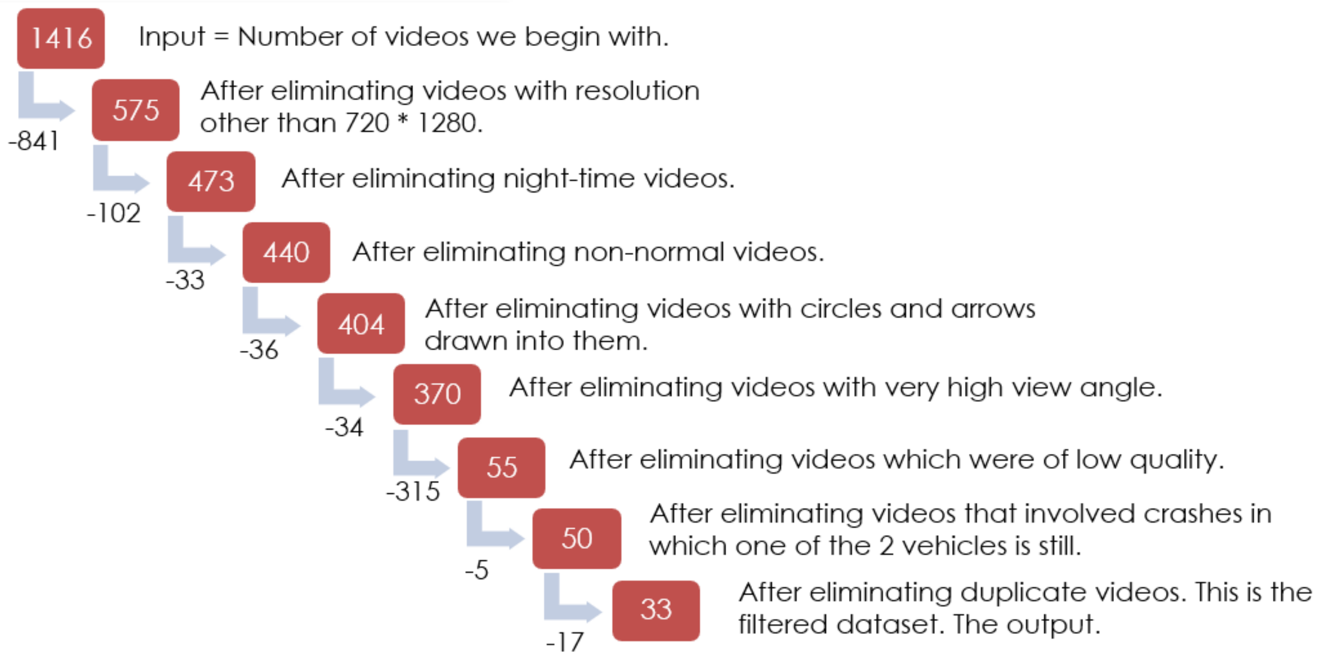
Open-source data, *Car accident detection dataset*, is obtained which was made available by (Shah et al., 2018).

The dataset contains 1416 accident videos, extracted frames, mappings, and graphs pertaining to the previously mentioned research paper. For the purpose of this study, data has been obtained in the form of videos.

Filtering

The following are criteria on the basis of which the data has been filtered to narrow down to a homogenous dataset,

1. **Resolution:** 720 * 1280
2. **Time:** Videos pertaining to daytime with sufficient light have been retained.
3. **Non-Normal Videos:** Footage is restricted to CCTVs. Dashcam videos were eliminated. Only crashes involving 2 cars have been considered in this study. Thus, all accidents involving humans, trucks, bikes, etc. were excluded.
4. **Noise:** Videos containing written text or shapes surrounding the area of crash have been removed.
5. **Angle of Camera:** Videos captured from a high perspective have been removed.
6. **Quality**: Videos having poor quality were removed.
7. **Duplicates**: The data consisted of duplicate videos which were eliminated to retain just one copy of each accident.

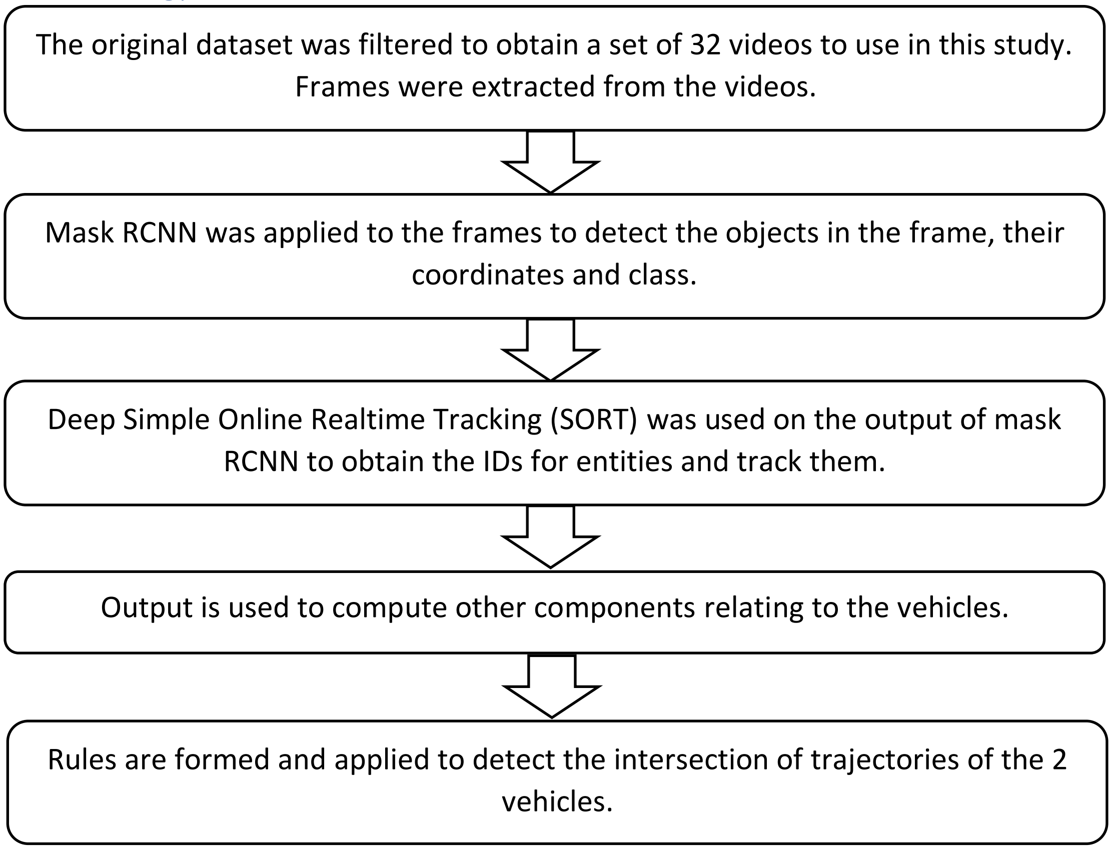

1416
Input = Number of videos we begin with.
575
After eliminating videos with resolution other than 720 * 1280.
-841
473
After eliminating night-time videos.
-102
440
After eliminating non-normal videos.
-33
404
After eliminating videos with circles and arrows drawn into them.
-36
370
After eliminating videos with very high view angle.
-34
55
After eliminating videos which were of low quality.
-315
50
After eliminating videos that involved crashes in which one of the 2 vehicles is still.
-5
33
After eliminating duplicate videos. This is the filtered dataset. The output.
-17

Post filtering of the data as discussed above; the data set comprises of 33 videos that are at an acceptable level of homogeneity.

It is observed that a dataset of 1416 is filtered down to merely 32 videos, which is a considerable loss of data. Here, this is necessary as the camera types are being filtered down to be homogenous. It is important to note that this filtering was mainly conducted on the basis of the camera that captures the video and not the accident itself. Further, only car to car crashes have been considered due to lack of homogenous data on crashes involving other types of vehicles and standalone crashes.

A test data of 10 videos has been obtained from YouTube.

Methodology



The original dataset was filtered to obtain a set of 32 videos to use in this study. Frames were extracted from the videos.

Mask RCNN was applied to the frames to detect the objects in the frame, their coordinates and class.

Deep Simple Online Realtime Tracking (SORT) was used on the output of mask RCNN to obtain the IDs for entities and track them.

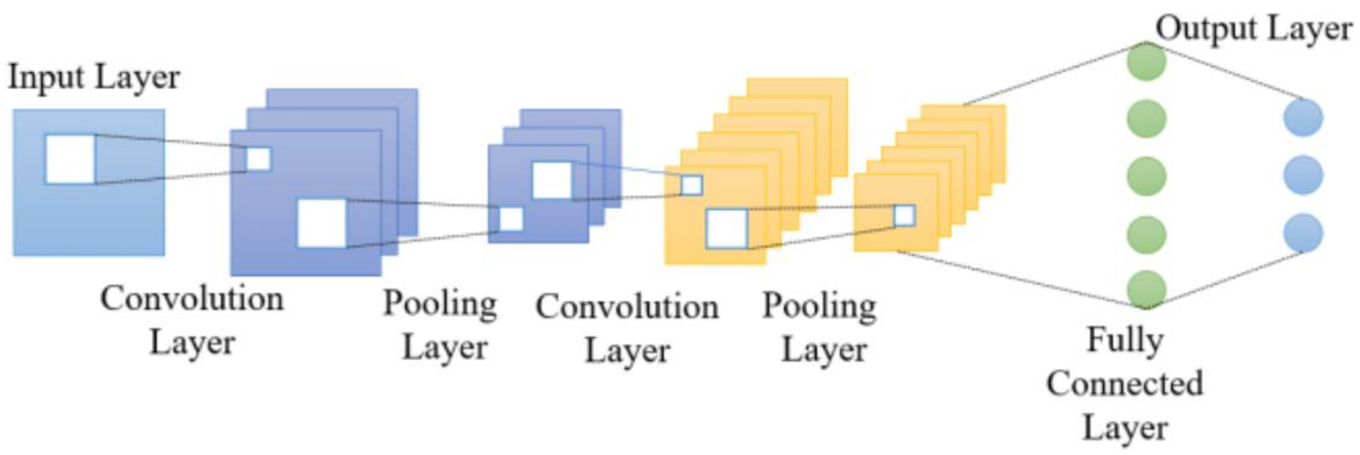
Output is used to compute other components relating to the vehicles.

Rules are formed and applied to detect the intersection of trajectories of the 2 vehicles.

Statistical Techniques

Convolutional Neural Networks

- CNNs are a variety of deep learning algorithm which are used to analyse images and videos. They work by breaking down an image into smaller pieces called "features", which are then used to classify or recognize objects in the image. In a nutshell, these algorithms look at a picture and identify the different shapes and colours in it. They are faster and more accurate than other methods.
- CNNs are made up of layers of neurons that process the image data in multiple ways. The first layer looks for simple features like edges and corners, while further layers look for more complex features like patterns and shapes. These layers work together to build a "feature map" of the image that can be used to make predictions about the different parts of the image.
- One of the key components of a CNN is the convolutional layer which uses a filter to scan the image and calculate a new value for each pixel based on its neighbours. This process is repeated many times with different filters to create a feature map of the image.
- Another important component of a CNN is the pooling layer which takes the output from the convolutional layer and down samples it by taking the maximum or average value within a small window of pixels. This helps to reduce the size of the feature map and prevents overfitting.
- Finally, the output from the pooling layer is fed into a fully connected layer, which performs the final classification or prediction. This layer takes the features learned from the previous layers and maps them to a set of output classes or values. The entire model is trained using a process called backpropagation, where the weights of the neurons are adjusted to minimize the error between the predicted and actual values.



Mask R-CNN

- Mask R-CNN is a type of deep learning algorithm that is used for object detection and instance segmentation in images. It uses a neural network to analyse images and identify different objects and can also draw a line around each object to show where it is.
- The network architecture of Mask R-CNN is like the original RCNN algorithm but adds a mask segmentation branch to the network. This branch is used to generate pixel-level masks for each detected object, which can then be used for more accurate instance segmentation.
- The mask segmentation branch uses a separate neural network to generate the masks. This network takes the features extracted from the input image and produces a mask for each object. The masks are matrices of the images where each pixel is assigned a probability of it containing the object.
- The training process for Mask R-CNN involves optimizing the network parameters to minimize the difference between the predicted masks and the ground truth masks. This is done using a combination of supervised and unsupervised learning techniques, such as the binary cross-entropy loss and the smooth L1 loss.
- Mask R-CNN is a state-of-the-art algorithm for object detection and instance segmentation and has many practical applications in areas like self-driving cars, medical imaging, and robotics. It can accurately detect and segment objects in complex images with multiple objects and backgrounds, making it a powerful tool for computer vision tasks.

SORT - Simple Online and Realtime Tracking

- SORT, or Simple Online and Realtime Tracking, is a type of computer algorithm that is used for object tracking in videos. It works by assigning unique IDs to each object in a video and tracking their movement over time.
- The SORT algorithm analyses the motion of each object to predict where it is likely to go next.
- The SORT algorithm is designed to be fast and efficient, which makes it well-suited for real-time applications like surveillance and robotics. It uses a Kalman filter to estimate the position and velocity of each object and updates these estimates as new information becomes available.
- Overall, the SORT algorithm is a powerful tool for object tracking in videos, and has many practical applications in areas like surveillance, traffic monitoring, and robotics. It can track objects in real-time with high accuracy and efficiency, making it a valuable tool for computer vision tasks.

Deep Sort

- Deep SORT is an advanced tracking algorithm used in computer vision which is designed to track multiple objects in real-time video streams, especially in a complex environment. Deep SORT uses a combination of object detection and tracking techniques, and it incorporates advanced mathematical models to achieve its high level of accuracy.
- At its core, Deep SORT has two main components: object detection and tracking. Object detection is the process of identifying objects in an image or video frame. In Deep SORT, object detection is mostly performed using a deep neural network-based object detector such as YOLO (You Only Look Once) or Faster R-CNN. These detectors work by dividing the input image into a set of regions, and then uses a convolutional neural network to classify each region as having an object or not. The regions classified as containing an object are then refined into bounding boxes that encloses them.
- Once the objects have been detected and localized, the tracking component of Deep SORT assigns a unique ID to each object and tracks its movement over time. For this this, Deep SORT uses a variant of the Kalman filter (mathematical model that predicts the position and velocity of an object based on its previous position and velocity) The Kalman filter is then used to estimate the state of each object and makes predictions about where the object will be in future.
- Deep SORT uses both appearance and motion features. Appearance features are based on the visual characteristics of the object, like colour, texture, and shape. These features are extracted from the bounding box around the object. Motion features are based on the object's movement, such as its speed and direction. These features are estimated using the Kalman filter.
- Deep SORT then uses a data association algorithm to match the detected objects with their corresponding tracks. The data association algorithm then uses a combination of appearance and motion features to determine which detections correspond to which tracks. This is done by calculating a distance metric which is typically a weighted sum of the Euclidean distance between the appearance features and the Mahalanobis distance between the motion features.
- Finally, Deep SORT updates the state of each object track based on the detected object's location and appearance features. It also removes tracks that are no longer valid, such as tracks that have been occluded by other objects.

Modelling

Video Processing

Extraction of Frames

After the filtering process, 32 videos were obtained from the open-source dataset. This is the input. From these videos still frames are extracted at 10 frames per second.

The following was considered when narrowing down on the precise rate of 10 fps. Firstly, the higher the number of frames per second, the slower the model runs as it deals with a higher load of data. Secondly, the distance moved by a vehicle in 1/10th of a second very small, hence data extracted at 10 fps is sufficient to model trajectories of vehicles and detect accidents.

Mask RCNN

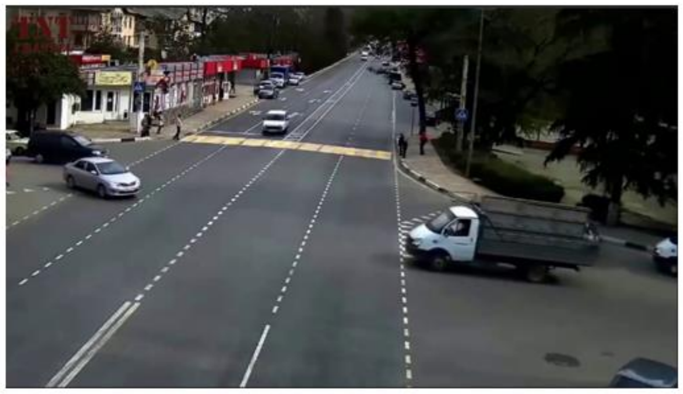
Mask RCNN is applied to the extracted frames of the filtered videos. The neural network processes the images to yield a numerical output along with the processed images. The images are characterized with the following,

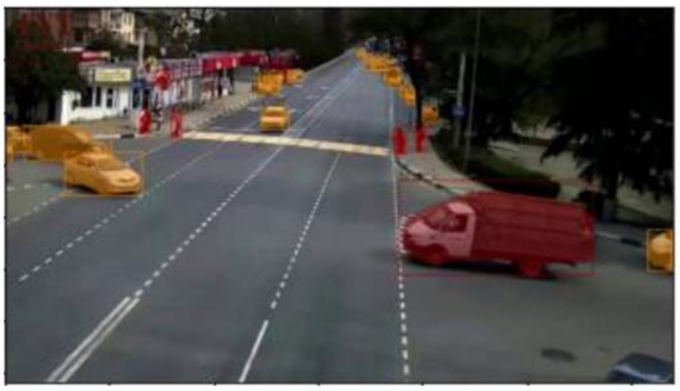
1. Mask RCNN yields a probability that it detects an object in the image. If the probability passes the pre-set threshold value (0.52 was found to be optimum in our case), then a “bounding box” is created for that object.

The box created encompasses the entity under consideration within its boundaries. The bounding box created is always parallel to the frame boundaries. Change in orientation of the vehicle will not change the orientation of the bounding box.

2. Mask RCNN will also output a 'mask' for each object detected in the image. The mask output gives a probability value to each pixel in the image. This is the probability that the object being identified lies in the given pixel. If the probability value is high enough (greater than the pre-set threshold of 0.59) then the x and y co-ordinates of that point are extracted for further use by our model. The result is a list that consists of pixel co-ordinates that have a high probability of being part of the object detected.

Following is an example,





The numerical output would encompass the following:

1. Classes of the entities detected. The class of the entity denotes the type of object detected. There are about 90 classes that Mask RCNN provides out of which only the following have been considered for this study: Car (class = 3), Bus (class = 6), Truck (class = 8).

2. Coordinates of the lower left vertex, the height and width of the bounding box of each entity.

3. Mask of each entity.

For the silver car observed in the left-hand side region, closest to the camera, in the above given picture Mask R-CNN yields the following output,

Frame	**Box x**	**Box y**	**Box w**	**Box h**	**Class**
2	154	281	153	101	3

Mask R-CNN yields a matrix for the mask points that

	203	204	205
292	0.375	0.754	0.631
293	0.595	0.782	0.886
294	0.666	0.569	0.801

The above table is converted to the following one in order to reduce the file size. The following table gives the coordinates of the pixels where the object lies.

Y	X
292	204

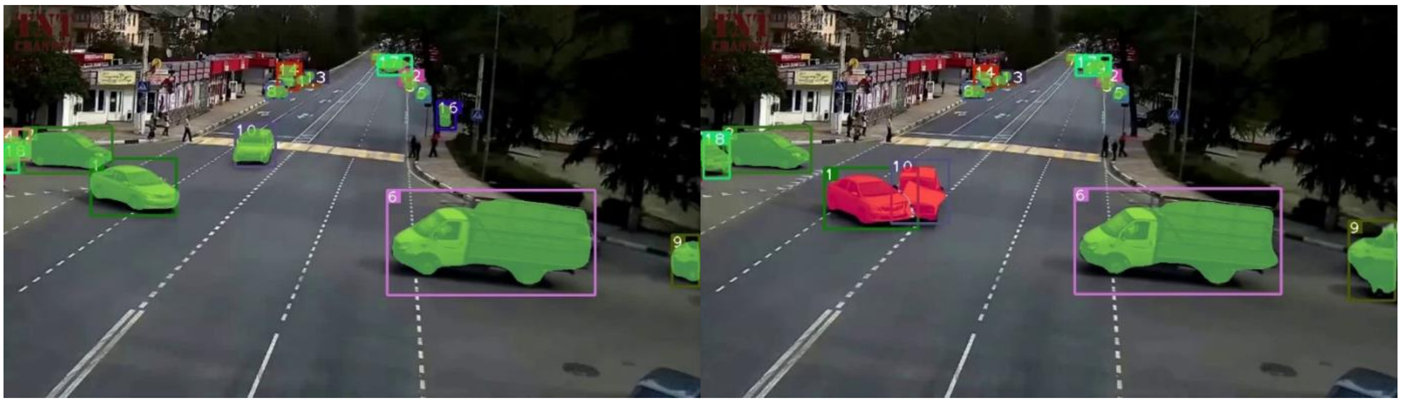
293	203
293	204
293	205
294	203
294	204
294	205

Deep SORT

Deep SORT is used to obtain IDs for the entities in the frames.

The study works on frames extracted from videos. A given entity in the video must have the same ID in all frames extracted from a single video, so that the algorithm can track the entity through subsequent frames. Deep SORT is applied to the output of mask RCNN to obtain the referenced IDs. This ID would help track the object as the study progresses.

Following is an example of the output obtained post applying Deep SORT.



Observe that cars with ID 1,6, and 10 retain the same ID across both frames.

A subset of the numerical output obtained for all the vehicles detected in the video, is as follows:

Frame	**ID**	**Box x**	**Box y**	**Box w**	**Box h**	**Class**
2	1	154	281	153	101	3
2	10	436	212	72	76	3
8	1	210	294	166	107	3
8	10	361	262	93	102	3

Computation of Components

The output of Deep SORT is now considered as input for the following.

The output of Deep SORT contains frame number, ID of the entity, x and y coordinate of lower left corner of bounding box, box width, box height and class of the entity. Using this, the following is obtained,

1. The coordinates of the centroids (x_c, y_c).

$$x_c = x\ co - ordinate\ of\ lower\ left\ corner + \frac{box\ width}{2}$$

$$y_c = y\ co - ordinate\ of\ lower\ left\ corner + \frac{box\ height}{2}$$

2. Length of the diagonal of the bounding box (using Pythagoras theorem).

$$Diagonal = \sqrt{(box\ width)^2 + (box\ height)^2}$$

If the calculated diagonal is found to be less than 75 pixels at for an ID in a frame or greater than 450 pixels, that ID is eliminated. If the diagonal is less than 75, then it means that the vehicle is too far to be identified accurately, and thus, isn't considered. If the diagonal is observed to be greater than 450 then it is an exceptionally long vehicle such as a large bus, or a carrier vehicle, whose dimensions lie beyond the standard length. Such vehicles have not been considered in the study.

The study then moves to calculating the slope and the distance between the centroids of the same ID in 2 successive frames. The slope is calculated as $\frac{y_2-y_1}{x_2-x_1}$ where the coordinates of the centroid in frame a are x_1, y_1 and the co-ordinates of the centroid for the same ID in frame b are x_2, y_2. The distance between the 2 centroids is calculated as the Euclidean measure, i.e., $\frac{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}}{b-a}$.

Normally the difference in frame number (b-a) is one but in some cases the vehicle is not detected for a few frames by mask RCNN, hence we take the average distance moved by that vehicle in one frame.

Using the slope calculated, the angle formed with the x-axis is found as follows,

$$m = Angle = \tan^{-1}(slope)$$

If the bounding box of the vehicle is within 5 pixels of the edges of the frame, then that observation is not considered further. This is because for a car that is entering the frame, the bounding box would only cover the part of the car that is visible in the frame. Hence, in the case of such cars the entire car is not captured in the bounding box and the displacement of the centroid would be reflective of their actual displacement. A similar effect is seen when the car is exiting from the frame. Thus, these observations are not considered.

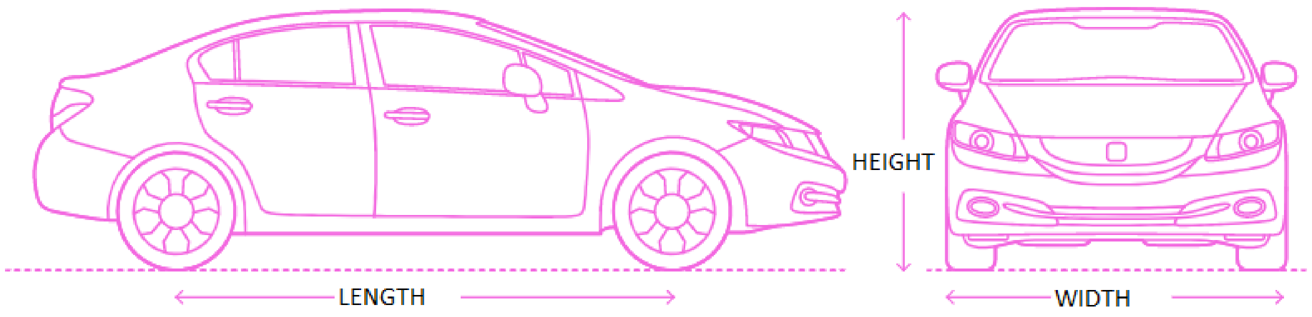
If the difference between the x co-ordinates and the difference between the y-co-ordinates of the 2 centroids both are found to be smaller than 2.5% of the diagonal of the respective bounding box, then the entity is classified to be stationary. Otherwise, the entity is said to be non-stationary. This is done due to instability of Mask RCNN. It produces slightly different boxes for the same stationary vehicle in different frames.

The previously calculated slope and distance measures are set be null for the stationary entities. For the moving entities, a moving average of order 3 is applied to the values of slope and distance. This is done to smooth out the values slightly and get rid of fluctuations in the values due to fluctuations in the boxes made by mask RCNN.

The values of the slope and distance for the first frame that is detected post a difference of more than 3 frames are not considered. For eg, consider ID 4 is detected in the initial frames until frame 23, and then it is not detected from frame 24 to 30. Then the slope and distance measures for ID 4 in frame 31 are not considered. This is because the slope and distance will see a sudden jump from frame 23 to 31.

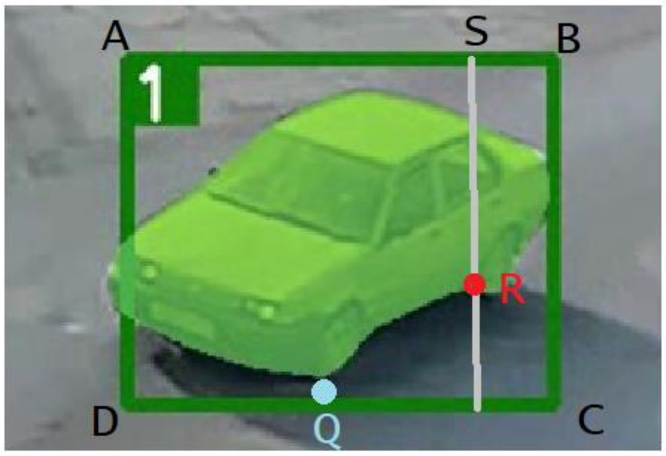
The next required component is speed. For the calculation of speed, a measure of distance per pixel is required i.e., the actual distance represented by the dimension of the pixel. Different pixels in the frame will represent different distances in the real world as some are closer to the camera and some are further back.

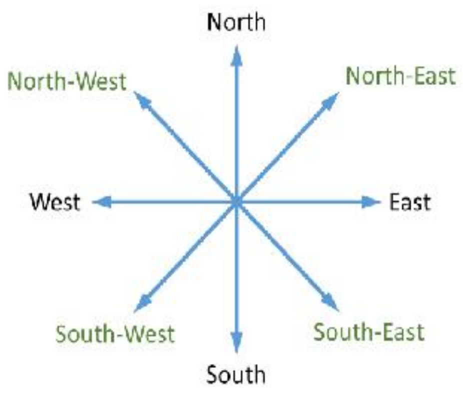
For the calculation of distance per pixel, the standard dimensions of the vehicle will have to be taken into consideration. Reference to the dimensions would be as follows,



Speed is obtained by the following steps,

1. Consider a car that is moving towards northeast or southwest, then the length of the car will lie in the right-side region of the bounding box (ABCD).





   If this is the direction, then we consider the right-most columns of pixels (columns between S and B) i.e., incorporating masking points that have a high value of x co-ordinates. 5% of the diagonal is the number of points that will be considered. If the diagonal is 100, then the 5 right-most columns will be considered. Out of these, the study requires the point which lies lowest on the masked region. This is the masking point (red point). Here, it is R, in the above figure.

2. It is proposed that R lies on the line which is the length of the car. Another thing to note is that the line will be in the same direction as the direction in which the car is travelling (which we earlier found as the slope). Hence the point found, R, in combination with the slope of the line enables us to determine the equation of the line of the length of the car.
   Using this equation, the co-ordinates of point Q are obtained. Point Q lies on the lower boundary of the bounding box.

3. The real-world length represented by QC can be obtained by using trigonometry. The standard length of a car or vehicle under consideration is known. The angle m was calculated before. Then,

$$Cosine(m) \times (Standard\ Length\ of\ Car) = QC$$

   The standard length will depend upon the class of the entity. [standard lengths]

4. QC is now known in real world distance terms and pixel terms. Dividing the values will yield the distance per pixel.

Similarly, if the vehicle is found to travelling towards northwest or southeast, then the same process is repeated by considering the left-side region of the bounding box.

If the vehicle is travelling towards one of the 4 main cardinal directions, QC i.e., the actual distance does not need to be calculated, instead it is considered as follows,

a. If the vehicle is travelling straight north or south (slope of vehicle is 90 degrees with the margin of 5 degrees), then only the width of the vehicle will be observed in the image, and this is compared to the standard width of the vehicle type.
b. If the vehicle is travelling straight east or west (slope of vehicle is 0 degrees with the margin of 5 degrees), then only the length of the vehicle will be observed in the image, and this is compared to the standard length of the vehicle type.

Using the distance per pixel, the actual distance between successive centroids can be obtained (product of pixel distance and distance per pixel). Using this, the speed is calculated as time is already known due to the FPS rate.

Output obtained from this section is as follows:

frame	**id**	**boxx**	**boxy**	**boxw**	**boxh**	**class**	**centx**	**centy**	**diagofbox**	**m**
3	1	160	281	157	104	3	238.5	333	188.3215336	0.260416667
3	10	425	217	74	79	3	462	256.5	108.2450923	-0.705
8	1	210	294	166	107	3	293	347.5	197.4968354	0.249278124
8	10	361	262	93	102	3	407.5	313	138.0326048	-1.203661327
…										

distinpixels	**angle**	**maskpointx**	**maskpointy**	**pixels**	**distperpixel**	**speed_m/s**
9.603691054	14.59657553	167	349	146.24	0.025443789	2.443542874
13.81358945	-35.18383662	492	268	46.71631206	0.067268736	9.292227014
14.96144353	13.9973094	218	364	157.428588	0.023698559	3.545646583
15.777803	-50.28024915	437	352	26.96958175	0.091105673	14.37447365
…						

Frame = Number of the Frame

ID = ID of every vehicle in the frame

Box x = x co-ordinate of the lower left corner of the bounding box

Box y = x co-ordinate of the lower left corner of the bounding box

Box w = Width of the bounding box

Box h = Height of the bounding box

Cent x = x co-ordinate of the centroid

Cent y = y co-ordinate of the centroid

Diagofbox = length of the diagonal

m = slope

distinpixels = distance between the centroids of the same ID in 2 successive frames

angle = angle between the slope and the x-axis

maskpointx = x co-ordinate of the masking point

maskpointy = y co-ordinate of the masking point

pixels = distance QC from above expressed in pixels

distperpixel = actual distance represented by each pixel in the frame

speed = speed of the vehicle

Trajectory for crash detection

In this project, the intersection of trajectories is used to detect crashes. It defines a collision as the intersection of 2 trajectories. Thus, tracing the path travelled and finding the intersection can help detect a crash. Now, the data is processed to obtain components that will aid the application of trajectory intersection.

A set of 7 frames is considered at each step. i.e., starting from the 8th frame, the current frame is considered along with the past 6 frames as a set (frame 2,3,4,5,6,7,8). Similarly, at the 9th frame, the set will consider frame 3,4,5,6,7,8,9. Then, a list is made of all the unique IDs observed in the set of frames. Then for each ID, in each frame maximum change in x coordinates and maximum change in y coordinates is noted.

If both maximum changes are found to be less than 5% of the diagonal, then the vehicle is said to move so less that a proper trajectory cannot be formed, and the car is considered to be still. The vertices of the bounding box are also taken into consideration. If the distance between the same vertex of the same bounding box in frame 1 and frame 7 is found to be less than 7 pixels, then the car is said to be stationary. The car is classified as being still if either of the above 2 conditions is satisfied.

For a moving vehicle the following is obtained, by considering sets of 7 frames (as discussed in the beginning of this section),

1. Beta Calculation
   $\beta = (X'X)^{-1}(X'Y)$
   i.e., the slope and intercept of the regression line fitted to the centroids of the entity through successive frames.
   This regression line is the trajectory. Following is an example.

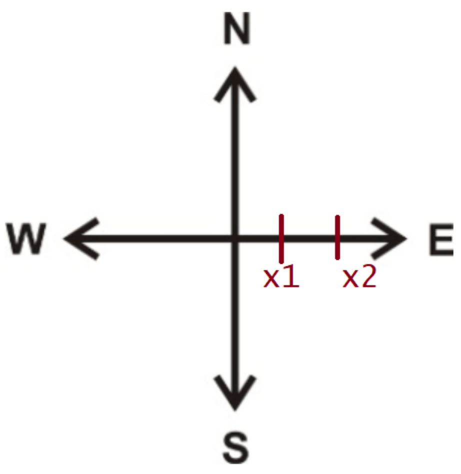
   Consider, an entity with ID 51 in video abc. Then the centroid co-ordinates of ID 51 in,

Frame number	**X coordinate of centroid**	**Y coordinate of centroid**
8	3.2	0.75
9	3.43	1.04
10	3.6	1.32
11	3.79	1.4
12	4.02	1.64
13	4.23	1.7
14	4.44	1.85

   $Equation\ of\ the\ Regression\ line\ would\ be\ Y = -1.887 + 0.8577 \times X$

   Thus, to obtain the trajectory of ID 51, a Y is regressed on X. This fitted line is the path travelled by ID 51 through the image frames of the video abc.

2. Direction Determination
   It is known that the vehicle travels along the fitted line obtained above. However, the direction in which the vehicle travels is not known. Thus, using the coordinates of the centroids a measure of direction is obtained as follows.



Everything in the frame is considered with reference to cartesian co-ordinates. First, we find the axis in which the vehicle is moving more by comparing the maximum changes.

If x coordinate of the centroid of the previous frame is less than the x coordinate of the centroid of the current frame, the vehicle is said to be moving to east, otherwise west. Similarly, the direction of North and South is determined using the y co-ordinates.

Consider the previous example. Here change in x coordinates is more, hence we conclude that the vehicle is moving mainly along the x-axis. The x co-ordinate in frame 12 is greater than that in frame 11. Thus, the entity with ID 51 is moving east.

The rule then moves further to find the coordinates point of intersection point. This is done by solving the regression equations of the trajectories of every 2 cars. Let i correspond to car i and j correspond to car j, then,

$$For\ car\ i, eq\ of\ the\ trajectory\ is{:}\ \ y = \beta_{01} + \beta_{11}x$$

$$For\ car\ j, eq\ of\ the\ trajectory\ is\ y = \beta_{02} + \beta_{12}x$$

The point of intersection is obtained by solving the above 2 equations.

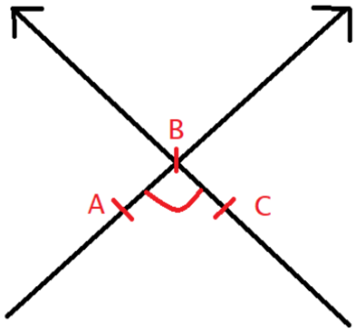
$$x_{intersection} = \frac{\beta_{02} - \beta_{01}}{\beta_{11} - \beta_{12}}$$

$$y_{intersection} = \beta_{01} + \beta_{11} \times \left(\frac{\beta_{02} - \beta_{01}}{\beta_{11} - \beta_{12}}\right)$$

Thus, the coordinates of the point of intersection have been obtained, where the accident is said to occur $(x_{intersection}, y_{intersection})$.

Once the coordinates have been found, the following is also determined,

1. Angle of Intersection



Suppose the 2 trajectories intersect at point B. Two points A & C are obtained on the respective trajectories using the line of regression and the direction they are travelling in. Using the co-ordinates of these 3 points, the angle between them (as shown in the picture) can be obtained. The formula is as follows,

$$Angle = \tan^{-1}\left(\frac{y_C - y_B}{x_C - x_B}\right) - \tan^{-1}\left(\frac{y_A - y_B}{x_A - x_B}\right)$$

To avoid, getting angles greater than 180 degrees,

$$Angle\ of\ Intersection = min\{Angle, 360 - Angle\}$$

2. Distance between the point of intersection and the last known centroids of both cars. Let us call this d_i and d_j.

$$d_i = \sqrt{(x_{intersection} - x_i)^2 + (y_{intersection} - y_i)^2}$$

Similarly,

$$d_j = \sqrt{(x_{intersection} - x_j)^2 + (y_{intersection} - y_j)^2}$$

3. Direction of the intersection point from the centroid.
Like last time, if $x_{intersection} > x_i$, then the trajectory is towards east. Otherwise, west. Here, $x_{intersection}$ is the x co-ordinate of the intersection point and x_i is the x co-ordinate of the centroid of car i. The same condition is checked for y coordinates, to get north or south.
If the direction of the intersection point from the centroid is same as that of the vehicle, then a value of 1 is assigned to that ID. Otherwise, 0.

4. S_i *and* S_j are the standardized values of the distance calculated above.

$$S_i = \frac{Distance\ between\ intersection\ pt\ and\ centroid\ of\ car}{Diagnol\ of\ the\ bounding\ box\ of\ car} = \frac{d_i}{diagnol\ of\ i}$$

5. The maximum speed and average speed for each ID is also noted.

6. Using average speed, a new variable Q_i is created.

$$Q_i = \frac{S_i}{Average\ speed\ of\ car\ i}$$

Conditions for Crash

Using the components calculated above, the following data frame is obtained. For all frames in the video, this data frame contains all pairs of vehicles detected in the frame. Now the task is to narrow down the rows in which a crash is occurring (A unique combination of frame and vehicle IDs involved in the crash).

frame	v1_id	v1_class	v1_maxspeed	v1_avgspeed	v1_samedir	v1_coef	v1_coef2	v2_id
8	1	3	3.545646583	2.626886917	1	0.305560705	0.116320464	10
8	9	3	6.008770199	1.20175404	1	7.198702554	5.990162975	10
9	9	3	6.008770199	1.544849831	1	6.607292034	4.276980133	10
10	1	3	3.545646583	2.287379828	1	2.932461294	1.282017642	9
…								

v2_class	v2_maxspeed	v2_avgspeed	v2_samedir	v2_coef	v2_coef2	intersection_angle
3	14.92601441	11.65811558	1	0.543503566	0.04662019	60.06367433
3	14.92601441	11.65811558	1	1.087542738	0.093286323	48.18998842
3	14.92601441	11.30010727	1	2.180655894	0.192976566	40.78424758
3	6.008770199	1.324156998	1	2.009392413	1.517488045	68.7349968
…						

vi_maxspeed = Maximum speed taken by that ID in the set of 7 frames

vi_avgspeed = Average speed of that ID in the set of 7 frames

vi_samedir = 1 means that both vehicles are moving towards the intersection, 0 means that at least one of the vehicles is not moving towards the intersection, -1 means that at least one of the vehicles is still.

vi_coef = S_i

vi_coef2 = Q_i

intersection_angle = angle at which the 2 trajectories intersect

S_i is the distance between the centroid of vehicle i and the intersection point divided by the diagonal of car i. In other words, it is a standardized distance. If this distance is too large, then the intersection point is far away from the vehicle, hence there is no crash. It was also observed that if the distance is too small, it meant that one of the vehicles was behind the other vehicle and there was no crash.

Hence, we set a condition that $0.229 < S_i, S_j < 0.63$. All other rows are filtered out.

Q_i is S_i divided by the average speed of vehicle i. It represents if the vehicle has sufficient speed to cover the standardized distance S_i. If the value of Q_i is too high, it means that the speed of the vehicle is too low, and it will not cover the distance in that particular frame and there is no crash occurring.

Hence, we set a condition that $0 < Q_i, Q_j < 1$. All other rows labeled no crash and filtered out.

Now we look at the samedir values. If samedir for both vehicles are 0, that means that they are moving away from the intersection point (and each other), the most common case being that they have passed each other at an intersection. Hence if both values are 0, that row is labeled as no crash.

After filtering out the rows labeled as no crash, it was observed that there were 2 common false positive cases in the rows left:

- One vehicle was passing the other (overtake).
- One vehicle was following the other.

To resolve these false positives, the series of distances between the 2 vehicles was considered.

$$D_{ijf} = \frac{Distance\ between\ the\ centroids\ of\ car\ i\ and\ j}{Diagonal\ of\ car\ i \times Diagonal\ of\ car\ j} \times 1000$$

This calculates the standardized distance between the vehicles.

This is calculated for the frame where the crash is detected, for the previous 10 frames and the following 10 frames. The set of 10 previous frames and the current frame is labeled as pre and the set of 10 following frames as post.

Then the slope of these distances is calculated with respect to the frames for the 2 sets. These are labeled as pre_{slope} and $post_{slope}$. If there are less than 3 values in the set, the slope is said to be 0. The absolute change between the 2 slopes is also calculated labeled as change. The following conditions are set using these variables (in the given order):

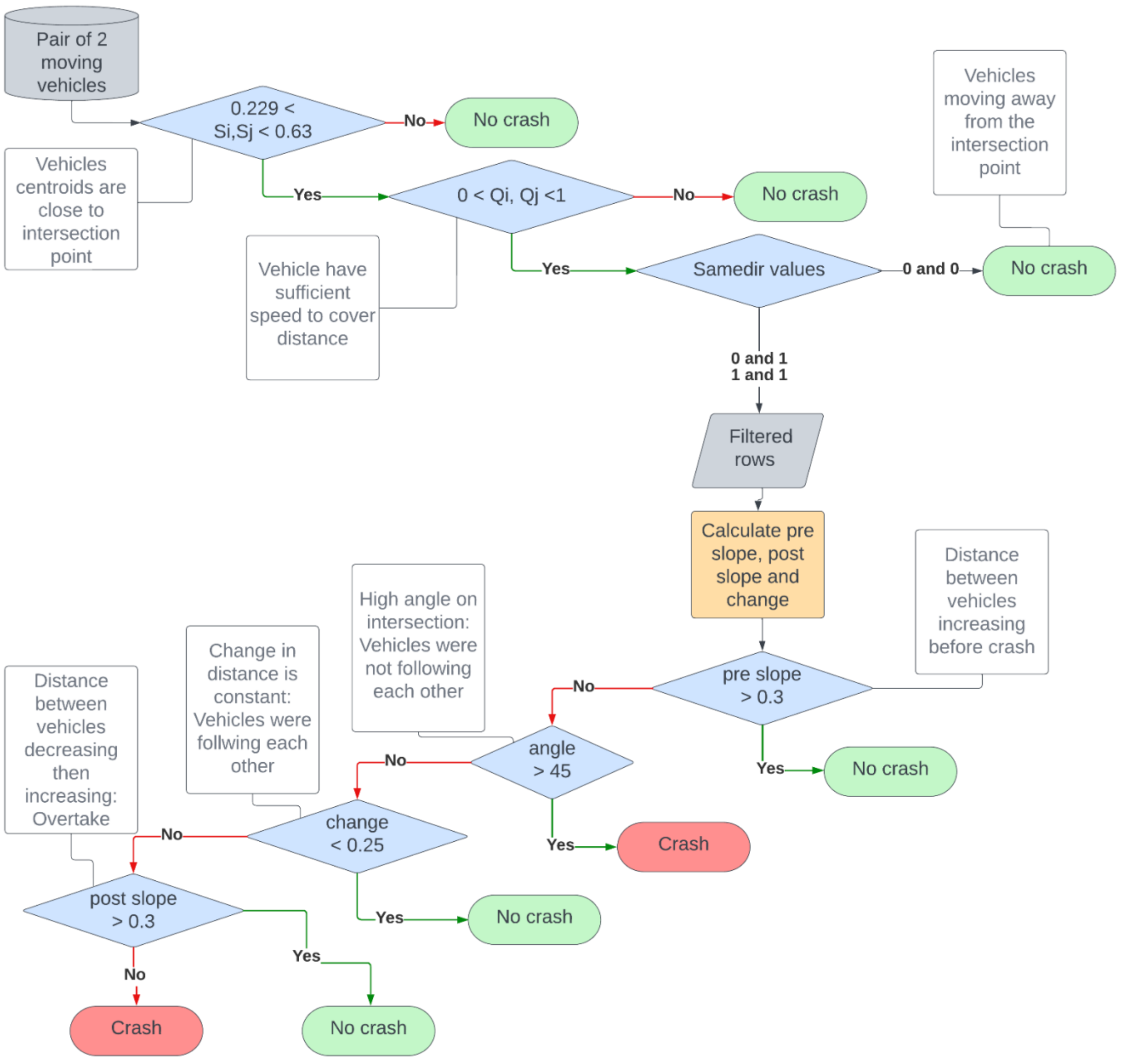
1. Intuitively, if the value for pre_{slope} is high, the distance between the 2 cars is increasing, and there is no crash.

Hence the rows with $pre_{slope} > 0.3$ are labeled as no crash.

2. If the angle of intersection between the vehicles is very high, then it indicates that the vehicles are not following each other.
   Hence, if angle of intersection > 45, the row is labeled as a crash.
3. Vehicles following each other will not have a significant change in the values of pre_{slope} and $post_{slope}$.
   Hence, if change < 0.25, the row is labeled as not a crash.
4. Finally, if vehicles are passing each other, or one vehicle overtakes the other, the distances in the post set will be increasing, meaning that the $post_{slope}$ will be increasing.
   Hence if $post_{slope} > 0.3$, the row is labeled as not a crash.
5. The remaining rows are considered as crashes.
   (Where $pre_{slope} < 0.3, change > 0.25, and\ post_{slope} < 0.3$).

Given the above conditions are satisfied, multiple crashes can also be detected in a single video.

Following is a process map representing the conditions used and the outcome:



Pair of 2 moving vehicles
0.229 < Si,Sj < 0.63
No
No crash
Vehicles centroids are close to intersection point
Yes
0 < Qi, Qj <1
No
No crash
Vehicles moving away from the intersection point
Vehicle have sufficient speed to cover distance
Yes
Samedir values
0 and 0
No crash
0 and 1
1 and 1
Filtered rows
Calculate pre slope, post slope and change
Distance between vehicles increasing before crash
High angle on intersection: Vehicles were not following each other
Change in distance is constant: Vehicles were follwing each other
Distance between vehicles decreasing then increasing: Overtake
No
pre slope > 0.3
Yes
No crash
No
angle > 45
Yes
Crash
No
change < 0.25
Yes
No crash
post slope > 0.3
Yes
No
Crash
No crash

Results

Classification Matrix:

		Actual	
		Crash	No Crash
Predicted	Crash	25	4
	No Crash	7	-

In each video, there are numerous pairs which are not predicted as crashes and are not crashes, hence there is no meaning in calculating True Negatives.

Overall, a good precision rate of 86.20689% is obtained with a recall of 81.645% approx.

It must be noted that the model is based on the 33 videos shown previously. Given the data set was small, there are chances that the model has been overfitted. Thus, the model is run on unseen data, i.e., test data.

		Actual	
		Crash	No Crash
Predicted	Crash	6	3
	No Crash	3	-

Overall, a good precision rate of 66.667% is obtained with a recall of 66.667% approx.

Deployment architecture and Turnaround time

For this study, it is proposed that the input videos are accepted in the form of 10 second clips, such that successive videos overlap for 2 seconds. Thus, a 10 second clip would be uploaded to the cloud every 8 seconds for further processing.

According to benchmarks set by published research papers, Mask RCNN takes 20 seconds and Deep SORT takes a further 5 seconds to process a 10 second video. The model then takes 2.5 seconds to apply the rules and detect a crash. Thus, the total time taken to detect a crash would be 20+5+2.5 = 27.5 seconds for one video of length of 10 seconds.

A new video shall be uploaded to the cloud every 8 seconds, meaning that 4 systems would be required to run simultaneously for smooth execution of this model. The first video will run on the first machine. As the 2nd, 3rd and 4th videos are uploaded after the said intervals of 8, 16 and 24 seconds, they'd be assigned to machines 2,3, and 4. This is because the 1st machine would still be processing the first video. When the 5th video is uploaded, it would have been 32 seconds since the first video was uploaded to the cloud. The first video would have finished processing and thus, the 5th video is run on machine one. Similarly, 6th video on machine 2 and so on.

Now, suppose a crash occurs. The clip would be uploaded after 10 seconds, after which it will get processed in the next 27.5 seconds to detect a crash. A margin of about 5 seconds is allowed for things like delay in upload, etc. Therefore, the crash is detected in 40-45 seconds approx. After it has occurred.

Limitations

- Image quality: Poor image quality or low-resolution images will affect the accuracy of the computer vision model. Our model is only trained on high resolution images.
- Time: The model does not work for night-time videos. The study is limited to daytime videos that have sufficient lighting. Changes in lighting conditions, such as glare, shadows or backlighting will also affect the accuracy of the model.
- Camera angle and position: The angle and position of the camera may not always be optimal to capture the necessary features for determining whether a crash has occurred. If the crash is to far away, it might not get detected accurately.
- Variations in crash types: There are many different types of crashes, each with their unique characteristics. Out model cannot thus capture unique car crashes.
- Environmental conditions: Environmental factors, such as weather conditions, can impact the ability of the camera to capture accurate images of the crash scene.
- Limited resources: Developing and implementing a computer vision model for crash detection can require significant resources, including funding, hardware, and technical expertise. Limited resources could limit the accuracy and effectiveness of the model in the real world.

- Static crashes and standalone crashes cannot be detected by this model. Detection of such type of crashes require the use deceleration and orientation, which has not been covered in this project.
- The run-time of the model is not significantly better than an actual witness at the accident site contacting an ambulance. However, the model would prove to be extremely helpful on highways, rural roads, and lonely regions, where the probability of somebody calling for medical help is very low. In such situations, the automation of this process would be of significant use.

Future Scope

- This model can be extended to incorporate the aspect of contacting a hospital or an ambulance when an accident is detected.
- Improved accuracy: One of the most significant future scopes for this project is improving the accuracy of the model by using other methods as well as training using more data.
- Expanded scope to include various times of the day, weather conditions, lighting, camera angles etc. which could be setup in any condition.
- Advanced feature recognition: As computer vision technology improves, it may be possible to develop models that can recognize more complex features, such as facial expressions, body language, or even specific makes and models of vehicles. For example, to detect intensity given a model is Toyota Fortuner versus Alto 800.
- Multi-camera analysis: In scenarios where there are multiple CCTV cameras available, future research could focus on developing computer vision models that can integrate and analyze data from multiple cameras simultaneously, to gain a more comprehensive understanding of the crash scene.
- Facial recognition-based crash detection: By incorporating facial recognition technology, the computer vision model could detect the facial expressions and emotions of drivers involved in a crash, providing additional insight into the cause of the crash.
- Integration with IOT devices and GPS which would provide additional information and real time mapping.
- The model can be extended to incorporate anticipatory crashes. This would mean that even if a crash is occurring just after the vehicles have exited the camera frame, the model would anticipate the crash occurring outside the frame based on the assessment done of the previous frames in which the vehicles were visible.

References

Shah, A. P., Lamare, J.-B., Nguyen-Anh, T., & Hauptmann, A. (2018, November). CADP: A Novel Dataset for CCTV Traffic Camera based Accident Analysis. 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). http://dx.doi.org/10.1109/avss.2018.8639160

Ghosh, S., Sunny, S. J., & Roney, R. (2019a, March). Accident detection using convolutional neural networks. 2019 International Conference on Data Science and Communication (IconDSC). http://dx.doi.org/10.1109/icondsc.2019.8816881

Chakraborty, P., Sharma, A., & Hegde, C. (2018, November). Freeway traffic incident detection from cameras: A semi-supervised learning approach. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). http://dx.doi.org/10.1109/itsc.2018.8569426

Ijjina, E. P., Chand, D., Gupta, S., & Goutham, K. (2019, July). Computer vision-based accident detection in traffic surveillance. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT). http://dx.doi.org/10.1109/icccnt45670.2019.8944469

Chang, W.-J., Chen, L.-B., & Su, K.-Y. (2019). DeepCrash: A deep learning-based internet of vehicles system for head-on and single-vehicle accident detection with emergency notification. IEEE Access, 7, 148163–148175. https://doi.org/10.1109/access.2019.2946468

Ojha, A., Sahu, S. P., & Dewangan, D. K. (2021, May 6). Vehicle Detection through Instance Segmentation using Mask R-CNN for Intelligent Vehicle System. 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). http://dx.doi.org/10.1109/iciccs51141.2021.9432374

Code

https://colab.research.google.com/drive/1fRYi-yFkY4gQHOo6ompBv0gqU0micltp?usp=sharing