1. What did you learn after looking on our dataset?

After observing the dataset, it can be seen that there are 4 different camera ids c10, c20, c21 and c23 each consisting of 126, 324, 146 and 484 images respectively. This will make a total of 1080 images. The images were clicked at an interval of 5min 43 seconds.

The images are taken from a parking structure and when there is no activity or movement in the parking lot (especially during night and mid-day), there seems to be multiple copies of the same image. During the day, when there is no human activity, there seems to be difference in the lighting conditions due to the sun as time passes. This can lead to redundant copies of the image with no new information for the network to learn during training.

The images in the dataset are of different dimensions namely,
(2688, 1520, 3): 12,   (640, 480, 3): 114,   (1920, 1080, 3): 950,   (10, 6, 3): 1,
(1100, 619, 3): 1,   (1200, 675, 3): 1,   'None': 1
Here the numbers represent **(height, width, channel): Number of images.**

It can be seen that there is one image of PNG format which is not an image. Hence, it is represented as None.

2. How does you program work?

The images in the dataset are a time series data taken from the cameras. Hence, they are continuous images taken in regular intervals. And since the image name consists of the timestamp, the images are in order and does not require sorting.

- Since the images are of different sizes, they are resized into one single set of target dimension. The calculation of the target dimension is shown in subsequent question.
- After resizing the image, two consecutive images are taken and the absolute difference between each pixel of the image is taken. The resulting image shows the difference between the two images.
- This image is then thresholded based on a value to highlight the major differences.
- After dilating the gradient changes in the image, we find the contours in the image that fit through these points.
- By finding the contour area, we then see how much the image changes with respect to its predecessor image. Thus, the score is calculated
- These steps are repeated for every image in the dataset. For image with score greater than 0, it means that there is a change from the previous image. These images are retained. For images with score equal to 0, the images are discarded since they are considered similar.

In the above steps, I have left out the preprocessing step done on the images like converting to Gray scale, Gaussian Blur etc. However, these operations are absolutely necessary and will be answered in the later part of this report.

3. What values did you decide to use for input parameters and how did you find these values?

There are 3 parameters that I considered relevant for this problem. They are,

- Target dimensions of the image (Since the size of each image varies)
- Minimum contour area
- Kernel size for Gaussian Blur.

One can also consider the Minimum Score to be a parameter. However, I have not considered this since tuning the value of "Minimum contour area" will encapsulate the changes of Minimum Score parameter.

I. Target dimensions of the image:

As mentioned in Question 1, the dataset contains images of different sizes. But for the algorithm to work, all the images need to be standardized into a single size. To calculate the optimal target dimension, I made sure to keep the Aspect ratio of the image a constant for majority of the images.
Here, I considered only the top 3 dimension with maximum number of images under them i.e., (1920, 1080), (640, 480) and (2688, 1520). Aspect ratio is calculated by dividing the height by width of the image.

For (1920, 1080): Aspect ratio = $\frac{1920}{1080}$ = 1.78

For (640, 480): Aspect ratio = $\frac{640}{480}$ = 1.33

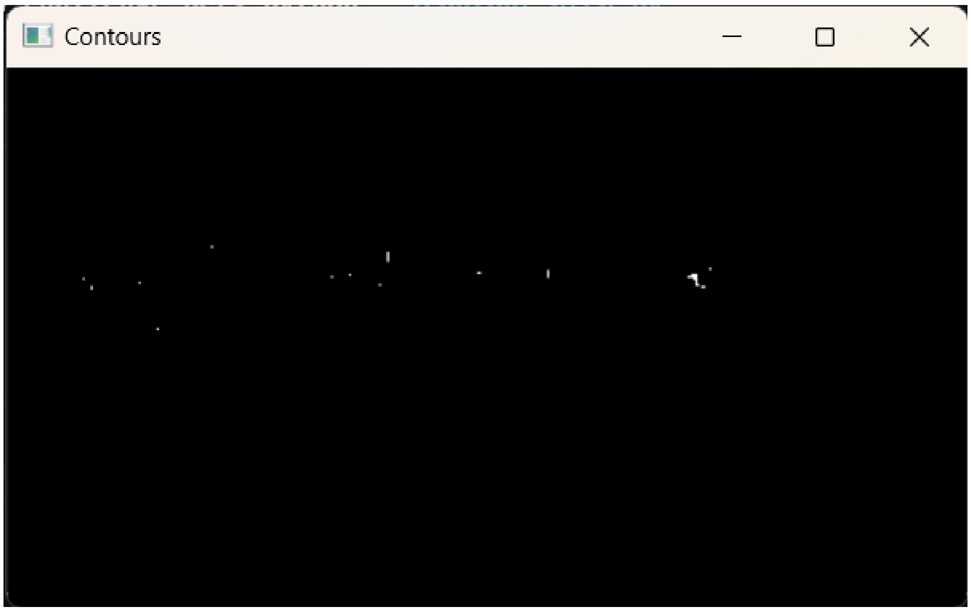
For (2688, 1520): Aspect ratio = $\frac{2688}{1520}$ = 1.77

It is clear that majority of the images have the Aspect ratio of 1.78. Hence, it would be ideal to maintain this ratio for all the images. This might affect the images of size (640, 480) as the images might get warped. However, these changes are minimal since its aspect ratio is very close to the ideal value of 1.78.

To calculate the dimension, I considered looking at the highest common factors of the (1920, 1080) with values less than (640, 480) since this will reduce the warping and distortions (The calculation is done for height and width separately. The highest factor of 1920 less than 640 is 480 for the height. For the width, the value is 270. This will maintain the ideal Aspect ratio.
Hence, the ideal target dimension for all the images is **(480, 270)**.

II. Minimum Contour Area:

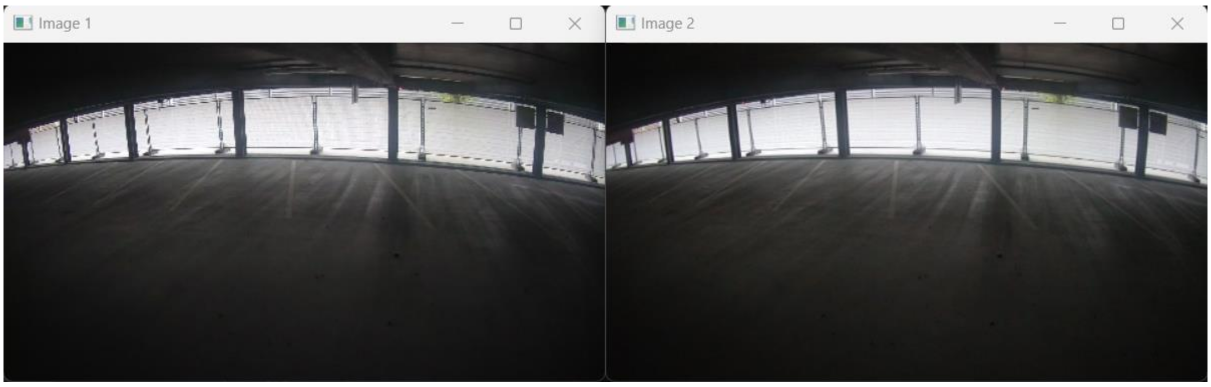
Consider the image below. It shows a Contour plot between two images. The white spaces represent the change between two consecutive images. Here, the change is very minimal. These kinds of changes are due to varying lighting conditions especially during the day. There is no human activity between the two images. Hence, the image copy can be ignored. The level of how much these contours need to be ignored is the function of the Minimum Contour Area parameter. This parameter acts a regularizer and ignores these minute changes. Fine-tuning of this parameter is shown in the later part.



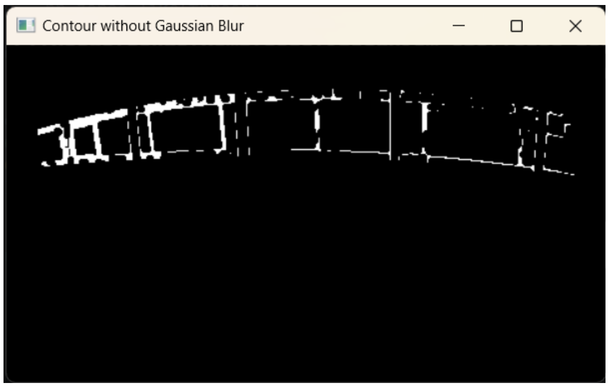
III. **Kernel size for Gaussian Blur:**

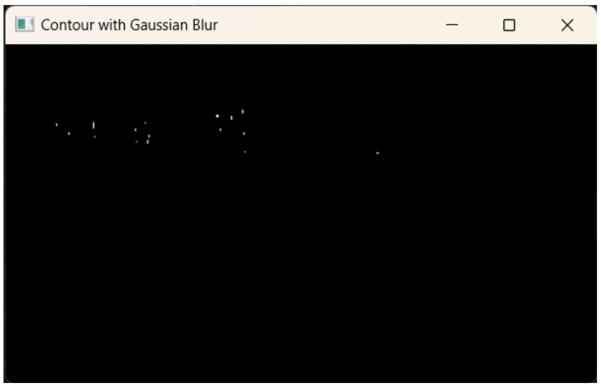
Since the images of different shapes are being resized, it is necessary to smoothen the images to eliminate minor changes between them. This will, to some extent, eliminate the changes in lighting conditions due to sunlight as time progresses. Hence, it is necessary to perform Gaussian Blur on the images. The kernel size of Gaussian Blur is a hyper parameter and is fine-tuned. An example of why Gaussian Blur is necessary is given as an example below.

Consider the image "c10-1623871098865.png" (dim: (2688, 1520)) and "c10-1623871124416.png" (dim: (640, 480)). After resizing both the images to (480, 270), the images are seen as follows:



The images to look similar to the naked eye and can be discarded. However, the algorithm will detect contours within these two images and consider them as two different images. But, by adding Gaussian blur to both the images, a smoothening effect occurs and this reduces the contour area. The Contour plot with and without Gaussian blur is shown below.





The left image indicates Contour without Gaussian blur and right one indicates contour with Gaussian blur of kernel size = 3. The kernel size can be varied as a hyper-parameter.

In my case, I used a Grid search technique to manually figure out the ideal parameter values for both Minimum Contour Area and Gaussian Blur kernel size. For Minimum Contour Area, I considered the values of [0, 100, 200, 500, 1000, 2000] and for Kernel size [0, 3, 5, 7]. I have evaluated the two values over the percentage of images retained after the execution of the program (The dataset contains 1080 images). I have made a table of these values and highlighted the cell with most ideal value for this dataset.

		Minimum Contour Area					
		0	**100**	**200**	**500**	**1000**	**2000**
Kernel Size	**0**	76.9%	64.3%	61.2%	53.7%	44.7%	37.1%
	3	66.7%	57.5%	52.1%	42.6%	36.1%	29.5%
	5	62.1%	54.6%	49.4%	40.3%	34.7%	27.8%
	7	58.2%	52.9%	46.7%	38.9%	33.2%	26.7%

The parameter value with Min. Contour area of 200 and kernel size of 3 is the ideal choice of parameter for this dataset. This parameter was considered for the following reasons:

- Since the target image size is (480, 270), larger kernel sizes like 5 and 7 eliminated images with human activities in the far background. We can observe this clearly when making a comparison of the images retained after running the algorithm.
- Same is the case for Minimum Contour area parameter. Larger values of this parameter eliminated human activities and small changes to car position in the background of many images. Hence, by both Qualitative and Quantitative analysis, (200, 3) was considered the ideal choice for these parameters.

4. What you would suggest to implement to improve data collection of unique cases in future?

The data collection can be improved for this dataset by following the steps below:

- The camera continued to collect data even at night time when there is less activity and changes. We can reduce the frequency of data collection at night time or at a time when the human activity is low (normally mid-day when people are already in their offices working) and increase the frequency during other times.

- During day time, it can be observed that the parking lot is in the same state with varying lighting conditions. The dataset contains too many copies of this and the light casts different shadows of the window as the sun moves. Even the current algorithm is not able to eliminate this scenario completely. Although, these types of images might be necessary during model training to make the model more robust, too many images can generate a bias and model may not generalize well. To avoid this, we can use motion sensors along with the cameras and only capture images when the motion sensors are triggered. Since motion sensors do not operate on visible light spectrum, changes in the lightings will not be affected. This will generate a good dataset with various lighting conditions for object detection.
- Try to obtain more images of people and cars during different time of the day/night. This will generate a dataset with different lighting conditions which is ideal for our task.
- Weather is also an important factor for our use case. Try to generate datasets during different weather conditions (summer, winter raining etc.).
- One more important thing to notice is in the given dataset, the parking lot was never completely full. It was mostly half empty or completely empty. Try to obtain images of the lot with packed cars for a more diverse generalization of the model.
- Perform a survey of the human activity timings in the parking lot. Weekday versus weekend activities, free v/s reserved parking spots etc., can give us information on what and where is the most ideal time to obtain data.

5. Any other comments about your solution?

Two images that stand out in this dataset is “c21_2021_03_27__12_53_37.png” with a size of (10, 6) and “c21_2021_03_27__10_36_36.png” which is not even an image. It is necessary to eliminate these bad images before using them for training. My current algorithm does not provide clear case handling for these images. However, it can be modified further.

Even though the images captured in this dataset are in time-series, there may occur scenarios where the same lighting conditions may repeat in irregular intervals. For example, after clicking 2 different images of the parking lot, the third image might look exactly like the first one. This kind of data is not handled in this algorithm.