Programming challenge - ML/ Computer vision engineer

Venkatesh Iyer

October 25, 2021

Exercise 1

What parameters have you decided to use for the provided example dataset?

Answer: The parameters I selected for this dataset are:

- Threshold - 45
- Gaussian smoothing - (21,21)
- Minimum contour area - 2000

  For these particular sets of values, it was observed that the algorithm was dealing with a balance between the duplicates as well as the noise present in the image frames.

Exercise 2

How you found these values?

Implementation: images.py

Answer: By analyzing the dataset, I found out that the images are timestamped sequence of a parking lot. Hence, to retain the temporal situations between the images, I compared a pair of frames to deal with replicas i.e 1st with 2nd, 2nd with 3rd, and so on.

In case, If do a one versus all comparison and remove the duplicate images from the dataset, we would lose the sequenced information contained in the images.

I performed my experiments by varying the Gaussian smoothing and the minimum contour area (the rate of change happening in between a pair) of sequenced images and removing the duplicates contained in them.

List of experiments performed:

	Gaussian smoothing	Minimum contour area	Duplicates found
Experiment 1	(5,5)	50	225
Experiment 2	(7,7)	50	255
Experiment 3	(11,11)	500	341
Experiment 4	(21,21)	500	363
Experiment 5	(21, 21)	2000	391
Experiment 6	(51,51)	2000	401
Experiment 7	(11,11)	2000	384

Experiment 1:
I started my experiments with a small set of values for both the smoothing factor as well as the contour area. Given sets of small values, the algorithm was very sensitive to changes happening in between the frames. A small contour area was responsible to capture very small changes happening and because of the low smoothing factor, the algorithm was also sensitive to noise present in the image frames.

The example below shows where both the frames are the same, even then it's detecting some noise and results in a difference between the frames.

By performing this experiment, I decided to increase the smoothing factor to reduce the noise the algorithm was capturing.

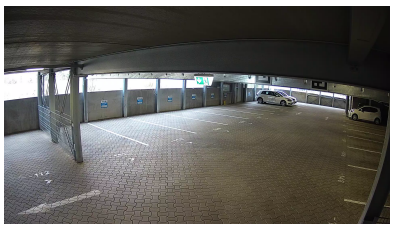


Figure 1: Previous frame

Figure 2: Next frame

Figure 3: Noise detected

Experiment 2:
In this experiment, I increased the smoothing factor whilst I kept the same contour area. I

observed the high-intensity pixels were being smoothened by the smoothing factor and the noise in the images was getting reduced. Also, due to smoothing, the number of duplicates was observed to be increasing.

The example below shows the comparison of the decrease in the noise being detected by the algorithm.

Figure 4: Smoothing factor: (5,5)

Figure 5: Smoothing factor: (7,7)

Experiment 3:
In this experiment, I increased the contour area to consider the bigger changes happening in between the frames. Also, increased smoothing factor results in fewer noises, bigger contour area neglects smaller changes happening and as a result, the duplicates are increasing.

Experiment 4:
Increasing the smoothing factor results in an increase of duplicates.

Experiment 5:
On analyzing the dataset, I observed that in a parking lot, the most important part is to keep the track of parking spaces i.e vehicles moving in and out of the parking space. Also, the company Kopernikus is working towards automated valet parking using AI and computer vision.

Given the above reasons, I decided to increase the contour area in order to only detect bigger changes (vehicles) happening in the frames. This results in an increase of duplicates.

Experiment 6:
In this experiment, I increased the smoothing factor exponentially. I observed that the algorithm then tends to overlook identifying the changes happening towards the corners of the frames. As it overlooks the changes happening in the frames, the duplicates tend to increase.

Figure 6: Corner changes

Figure 7: Overlooked detecting the corners

Experiment 7:
Keeping the contour area constant whilst decreasing the smoothing factor again introduces some noise identification in between the frames. Hence, we can see the duplicates increasing in number compared to experiments 6 and 5.

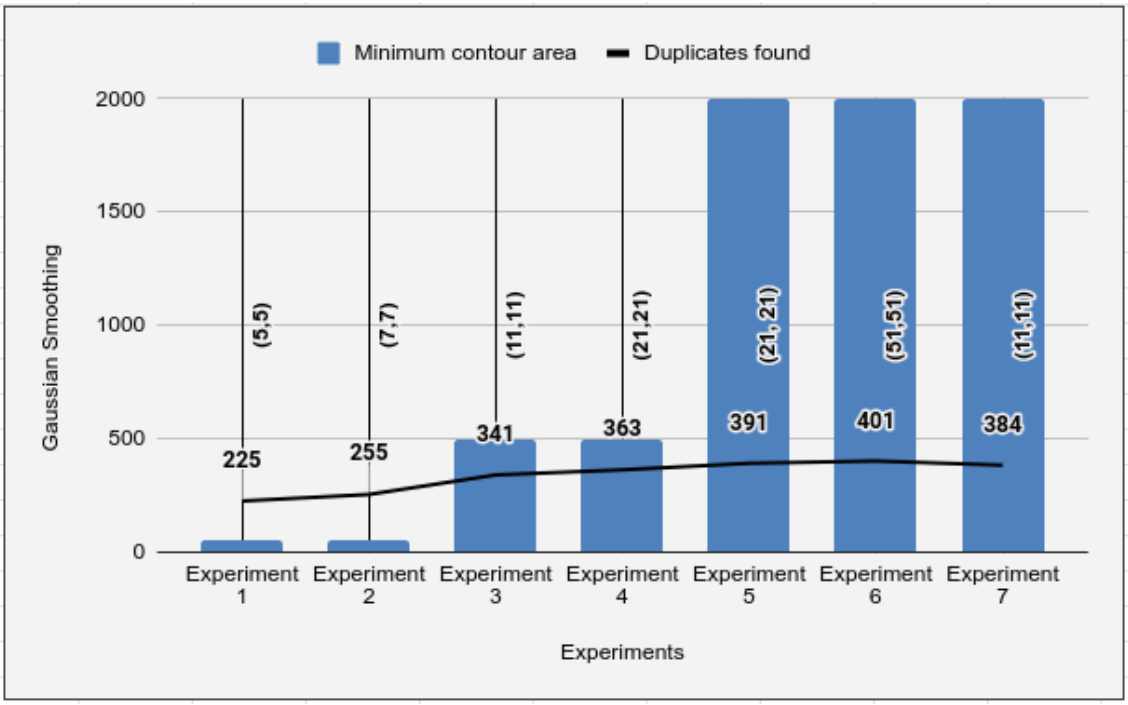
After all these experiments, I conclude that a smoothing factor of **(21,21)** and a contour area of **2000** maintains a balance between noise and the changes happening in between the frames.

However, if we exclude the counter area and match the frames pixel-to-pixel (absolute difference of pixels between two frames), it gives us the maximum number of duplicates found. i.e **408**.

Exercise 3

What amount of duplicates script found with these parameters?

Answer: I have summarized my readings using a simple graph plot. The graph shows the relation between contour area, smoothing factor, and the resulting number of duplicates in the dataset.



Exercise 4

What you would suggest improving to make data collection of unique cases better?

Answer: Dataset collection improvisation depends completely on the use case. According to my, there are two kinds of scenarios from the above-given dataset.

1.) Contrast changes are not important and we are more interested in vehicles moving in and out from the parking lot. But, with this technique of removing duplicates from the dataset, the algorithm detects the change in lighting conditions in between the frames and considers this as a change in the image frames. This can cause some problems if we are interested in only spotting vehicles.

For this kind of scenario, contrast invariant color space split might be of help.

The example below shows how lighting conditions affect the processing of the algorithm.

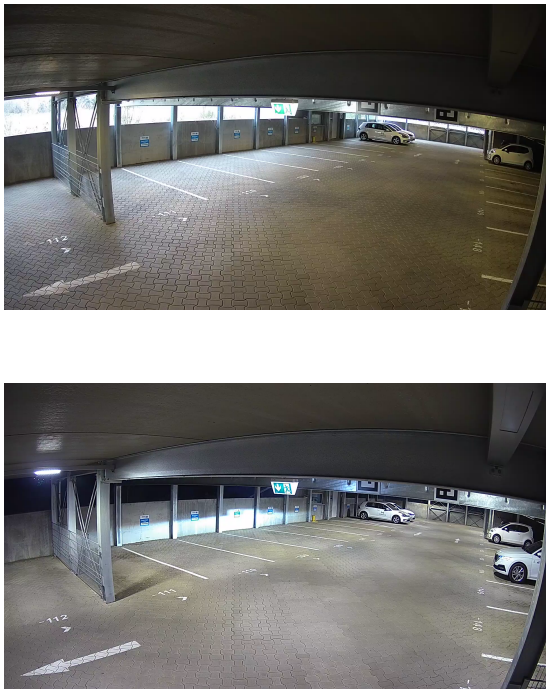


Figure 8: Original frames

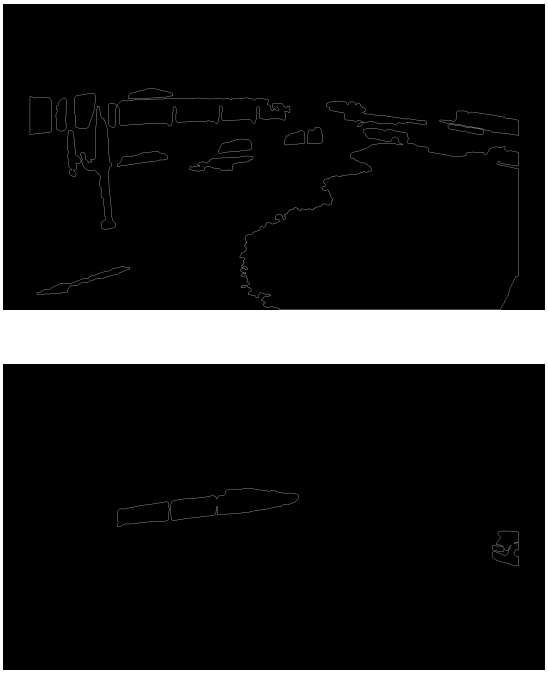


Figure 9: Captured differences because of lighting

Implementation: hash.py

2.) Cases where contrast or lighting conditions are important to be noted. In such cases, using hash values of images is one of the solutions wherein hashing techniques like average hashing, difference hashing can work in order to detect the changes in lighting conditions.

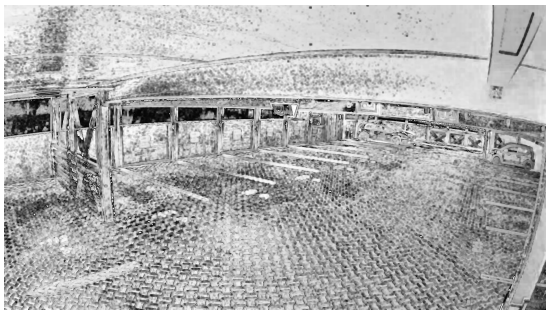
However, when I implemented it, I observed that luminance or lighting changes are identified majorly. But, it doesn't work well with removing the duplicates from image sequences as the majority content of the image matches with each other.

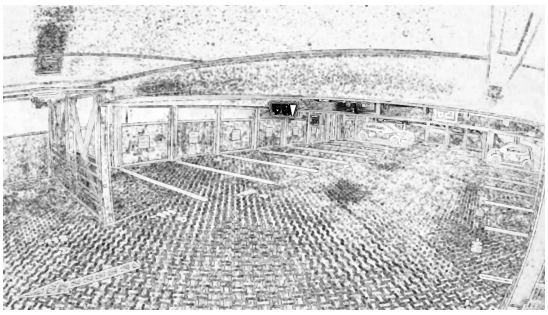
Implementation: ssi.py

3.) In order to match the structural similarity between the frames, I used Structural Similarity Index (SSIM) measure. I observed as the frames are majorly similar to each other, the score of

SSIM always is in between the range of **0.85 - 0.95** while **1.0** being a complete match. The SSIM value drops around **0.60 - 0.65** whenever it detects any changes in between the frames. However, even SSIM is sensitive to lighting and illumination changes. Given this reason, finding duplicates becomes difficult.

The example below shows how the lighting effects the processing of the algorithm





Implementation: without_preprocessing.py

4.) I also tried spotting the differences in between the frames without doing any pre-processing on the images. I used Canny edge detector to find the contours of difference in between the frames. However, I observed, this does not give any drastic positive results in terms of excluding the illumination of lighting changes.

To summarize, the individually given and implemented algorithms work well if we do not exclude the lighting effects and consider only the moving objects (in our case vehicles). In the case of excluding the lighting effects and identifying the changes happening in between the frames, I believe a combination of the above-given algorithms would suit such use cases.

Exercise 5

Any other comments about imaging_interview.py or your solution?

Answer: Solutions provided can always be improvised as per different use cases and can be implemented in a more efficient way.

Depending on the use cases, a combination of multiple algorithms can be tried to attain better results.