Imaging Interview Answers

kushaln070@gmail.com

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

   Ans:

- The dataset consists of images captured from different cameras mounted inside an enclosed parking lot with different aspect ratios. The objects in an image could be cars, pedestrians, or other objects typically found inside a parking lot. The lighting of the parking lot can come from 2 sources: one from an **active source (parking lot-mounted lights and vehicle headlights)** and the other from a **passive source (sunlight)**.
- Illumination from parking lot-mounted lights is constant and illumination from sunlight is varying and dynamic throughout the day (due to clouds and position of sun).
- Filtering out **similar-looking** images by **image differencing** method is **effective** when only parking lot light is present (during the night). This method seems **ineffective** when there is varying and dynamic sunlight illuminating the parking lot.
- Approximately 10% to 30% of the image is covered by the ceiling of the parking lot in a few **mounting positions,** which results in less coverage of the useful area.

2. **How does you program work?**

   Ans: The program can be found in this GitHub link:

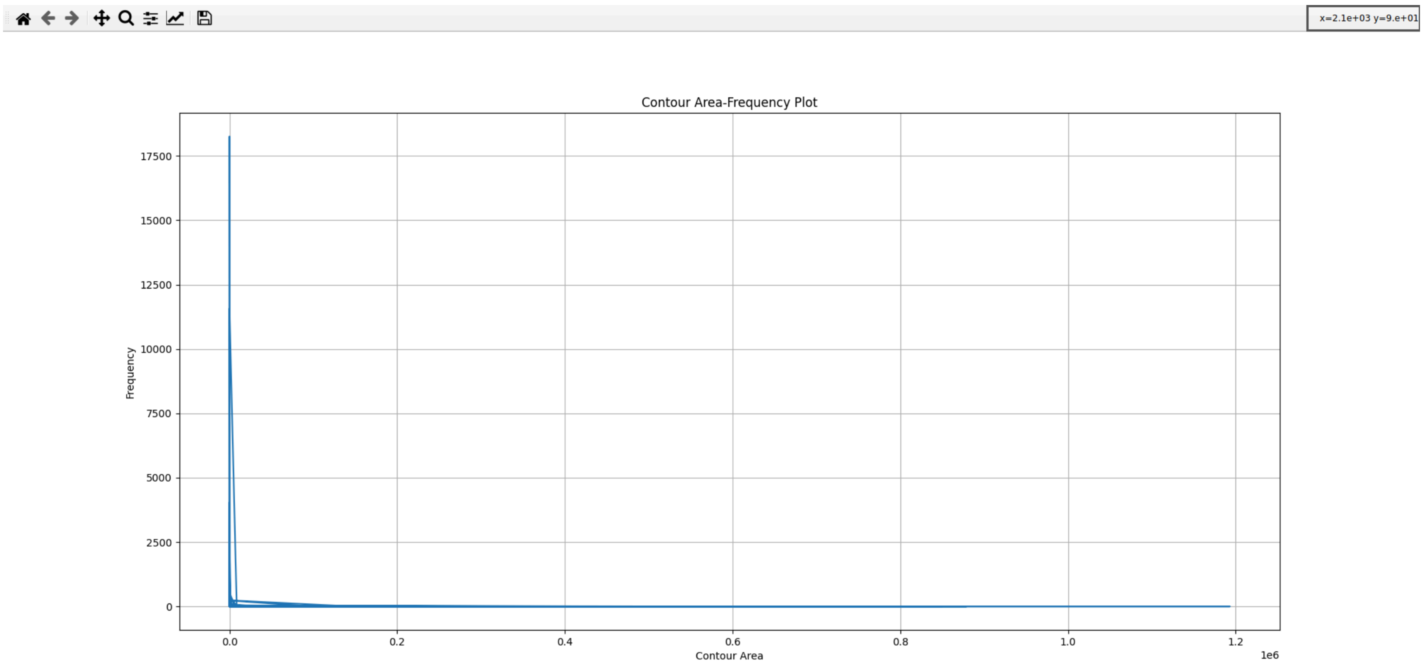
- git@github.com:kushalnarasimha/kopernikus_interview.git
- The python file **imaging_interview.py** takes in a **path to the folder** containing the captured images as **input** and creates **new folders** by the name ***camera_id***. Inside a *camera_id* folder essential images are stored. If the python file is run with an argument flag "*–store_non_ess_frames*", non-essential images will be stored inside a folder name (*store_non_ess_frames*) in the same input path.
- Control flow:
  1. Images in the input *img_files_fold_path* are sorted using *sort_images()* based on the *camera_id* and *timestamp*. This function outputs a dictionary where keys are *camera_id* and values are corresponding image file names sorted based on their timestamp.
  2. Loop over the dictionary (key=*camer_id* and value=*image_file_names*) to filter out non-essential images.
  3. Inside a loop, the image in the 1st iteration is automatically stored and considered as prev_img. In the next iteration onwards, the ***current_img*** and ***prev_img*** are preprocessed using *preprocess_image_change_detection()*. The kernel *radius=5* is used for *cv2.GaussianBlur()* and *black_mask* tuple value is obtained using *get_black_mask()* based on the *camera_id*. Then the function *compare_frames_change_detection()* with *gray_current_img*, *gray_previous_img* and ***min_contour_area=1500*** as input parameters is used to find how much images differ from each other. One of the results of the function *compare_frames_change_detection()* is *score*. *score* is a sum of areas of the contours which are greater than the *min_contour_area*. If the *score > 0*, the *current_img* is considerably different than that of the *prev_img* and ***cuurent_img* is considered as an essential frame**.

Else there is no difference and ***curent_img*** **is considered as a non-essential frame.** Then *prev_img=current_img* is made and loop repeats.

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

Ans: The input parameters are: ***gaussian_blur_radius_list = [5]*** and ***min_contour_area=1500.***

- Using *cv2.GaussianBlur()* on images before *cv2.absdiff()* and *cv2.dilate()* operation is effective because it eliminates dilation of very small areas caused by few high intensity pixels inside a low intensity pixel region. The gaussian blur radius 5 is optimal with regards to smoothing effect as well as computation reduction. Blur radius greater than 7 seems more smoothing and causes improper contour boundaries around an object.
- The value *min_contour_area=1500* is found optimal to all objects (pedestrians and cars) at different scales or sizes. At first, the Histogram plot of all contour areas in all frames is plotted as shown below. A reference of *min_contour_area*=2000 is taken to eliminate most minimum area contours. Later specific cases are visually inspected to find suitable *min_contour_area.* After inspection, *min_contour_area=1500* is found optimal for both small objects and large objects.



Imaging Interview Answers

kushaln070@gmail.com

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

   Ans:

   - Machine Learning methods: Training **Siamese Networks** to predict whether the 2 input images are similar or dissimilar. Siamese Networks can encode more diverse features in an image to predict similarity.
     Context-Aware Scene understanding models can encode different features in an image to characterise and describe an image by number of objects, types, day or night etc. This context of the scene can later be used to filter out similar-looking frames.
   - **Illumination invariant feature descriptor**: Feature detectors and feature descriptors can be used to find a match between 2 images. Feature descriptors that are invariant to changes in illumination can be effective in filtering out non-essential images, especially during the presence of varying and dynamic sunlight.

5. **Any other comments about your solution?**

   Ans: My solution works very well in all situations to filter out non-essential frames. It filtered out similar-looking frames even during the presence of less varying sunlight. However the solution is ineffective when there is huge variation in sunlight between frames. I consider having **few similar-looking frames with different illumination from sunlight** is useful for training or evaluation of object detectors. Such that the trained object detector becomes robust against varying sunlight.