Programming challenge for Perception team member in Kopernikus

Questions:

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

Answer:

The images for each camera is being recorded at different time stamps throughout the day/night. The cameras used for collecting data are located in the parking lot. From the observation of the images of different cameras. It looks like the camera-ids: - c20, c21 and c23 are located on the same floor. There is no clear evidence that camera c10 belongs to same parking lot of the above mentioned cameras.

The images of all camera-ids contain lot of duplicates or non-essential data. There is definite need for pre-processing dataset.

The following are the important observations which has to be correct during the optimization of dataset are:

1. Duplicates and similar looking images.
2. Corrupted or modified PNG images.
3. Size of the image- few images have different sizes (H*W)

2. How does your program work?

Answer:

My program which I provided, defines a class `OrganiseDataset` using a `dataclass` decorator. The `OrgansieDataset` class is responsible for removing duplicate or similar-looking images based on the specified parameters.

Here's how the program works:

1. The `OrganiseDataset` class has a `folder_path` attribute, which represents path to the folder containing the images.

2. The `remove_duplicates` method is the main entry point of the program. It takes parameters for `gaussian_blur_radius_list`, `min_contour_area`, and `min_score`, which define the image preprocessing and comparison criteria. It removes duplicate or similar-looking images based on these parameters.

3. The `get_camera_id_indices` method is responsible for obtaining camera ID indices for the images in the folder. It returns a tuple containing a dictionary and a list of Image files. The dictionary maps camera IDs (based on prefixes) to a list of indices representing the positions of the corresponding filenames in the list. The list contains the filenames sorted in ascending order.

4. The `remove_ids` method removes the identified non-essential files based on the given dictionary of camera IDs and indices. It uses the `os.remove` function to delete the files from the specified folder path.

5. From the `imaging_interview` module the functions `preprocess_image_change_detection` and `compare_frames_change_detection` is used to perform preprocessing and comparison between two frames.

```
function remove_duplicates():
    Get the camera ID, indices and filenames from the folder

    Initialize an empty dictionary to store non-essential files

    for each camera ID and its image indices:
        Initialize an empty list to store remove IDs

        for each image index:
            Get the file path based on the folder path and filename
            Read the image using cv2.imread

            if the image is None:
                Add the image index to the remove IDs list and continue

            if it is the first image:
                Pre-process the image and store it as the previous frame
            else:
                Pre-process the image and store it as the next frame

                if the shape of the previous frame and next frame are different:
                    Resize the previous frame to match the shape of the next frame

                Compare the frames and calculate the score

                if the score is below the threshold:
                    Add the image index to the remove indices list
                else:
                    Update the previous frame with the next frame

        Add the camera ID and remove indices list to the non-essential files
dictionary

    Remove the non-essential files for each camera ID using the remove indices
```

Pseudo code of the program

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

Answer:

The values for the Gaussian_blur_radius_list: [3,5,7]

From the documentation of opencv, the values of kernel size must be positive and odd. The Center of the kernel is not possible if the size of the kernel size is even. The others reason is:

i) Computational cost: Smaller kernel sizes (3,3) are computationally more efficient.
ii) Smoothing effect: Romiving noise and reducing high frequency is better with (7,7) kernel size.
iii) The cumulative effect of each radius 3,5,7 gives effective image smoothing and better results.

The value of min_contour_area: 2000

The images are from the different cameras of a parking lot. Considering the fact, the vehicles such as Cars, SUVs are the frequent objects appearing in an image. The average pixels of a normal car in an image would be around 10000 pixels (assumption). Then, the min_contour_area is 20% of 10000 is 2000. Even if there are two cars this value would be sufficient to filter out the images which don't have duplicates after the comparison with the prev_frame.

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

Answer:

The methods which improve data collection in scenarios such as a parking lot with different time stamps are:

1. Implementation of Image hashing and Image comparison algorithms.
2. Filter images based on time, such as excluding images for a specified time-window when the parking lot is closed or there is no change in the state of a vehicle or object in the parking lot.
3. Use of Machine learning algorithms such as CNN, Siamese Networks for filtering duplicates and non-essential data.

5. Any other comments about your solution?

Answer:

The `OrganiseDataset` class is used to solve the problem of removing duplicated and similar looking images.

My idea was to store the `camera_ids` and list of `indices` of file_path for each `camera_id` in a dictionary. So that during comparison of frames it is clear that we are comparing on the frames of same `camera_id`. The indices of repeated images of each `camera_id` are stored separately, which can be used to remove repeated images. This can also use to generate statistics of particular camera and how the input parameters effect the results.

The input parameters to `remove_duplicates` function can be changes based on the requirements. The Unit test to verify `camer_ids` can be used for testing and also for integration in CI/CD workflows.