Detecting lens Smear

Senthil Palanisamy
Jordan Zeeb

What is Lens Smear?

-A smear on a camera lense causes distortion and occlusion of its images.

-A smear is usually caused by dirt and oil smudged onto the lens.

-Certain areas in an image that are purposely transformed to remove obscuring objects are not considered a smear.

Our Methodology

1. Method 1 - Smear is an area of uniform smoothness across a sequence
2. Method 2 - Smear is a dark and nearly constant region
    a. Using a known and fixed threshold
    b. Using Adaptive threshold
3. Method 3- Smear is the only area in the image, which has nearly zero optical flow (The best method with best results and has very weak and realistic assumptions about the data)

Method -1 - Smoothness based.
Core idea and observations

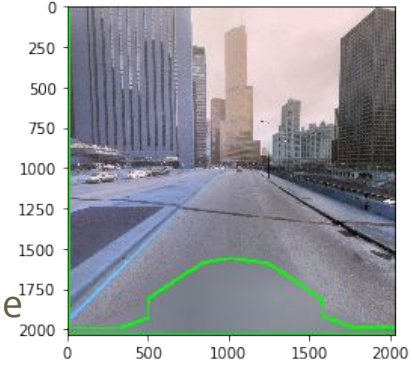
1. The primary intuition behind the formulation of this method is the fact that lens smear are very smooth and hence, one way to define lens smear for the purpose of a computer vision algorithm could be **“An area of pixels patches which remains uniformly smooth across an image sequence”**
2. Armed with a useful definition of a lens smear, we could just detect gradients of the image and spot areas where the image gradients are zero
3. But this is not sufficient since an image always has some smooth regions. The sky for example is always smooth. But as the vehicle moves, all the pixels in the images change their image gradient but there is only one regions which doesn’t change the image gradient and it always remains smooth: The area where lens smear occurs.

Method -1 Detailed explanation

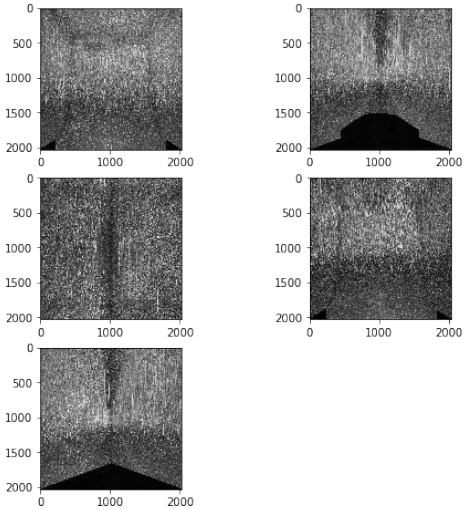
1. **Image preprocessing:** Do histogram equalisation to enhance contrast
2. **Core algorithm:-**
    a. Calculate image gradient on each image
    b. Calculate the max image gradient of all pixels for an image across a given image sequence
    c. Threshold the grey scale image to binary image (threshold for image binarization is taken to be a very small value (5). This specific value of thresholding in this case doesn't account as overfitting to data. Since we are interested in finding pixels which didn't posses any sort of gradients across the whole sequence, this thresholding value on the a max gradient image is in fact valid assumption and not a overfitting to the data.
    d. Invert the image to obtain binary image
3. **Post processing:-**
    a. Perform morphological closing (The net effect is to connect neighbouring pixels in a point cluster without expanding the boundaries of the cluster)
        i. Dilate the image to join disconnected regions.
        ii. Erode the images to eliminate the overgrowth of boundaries
    b. Detect contours on the binary image and draw the contours on the color image

Downsides of method 1

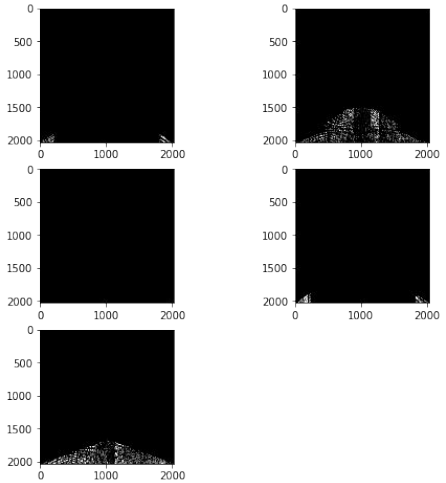
1. The biggest downside of method-1 is its own definition. Some of these image sequence contains a portion of blurred regions, which are not actually lens smears but these regions get noticed as lens smear due to their uniform smoothness across the image
2. If this blurred region is known in advance, we could remove this by maintaining a mask where a blurred region is to be expected.
3. But we did not remove this in our algorithm, since we wanted to show the false positive cases of this method
4. But since some of these artifacts seem to be purposely introduced, probably to obscure some proprietary data, its reasonable to assume that we have a known mask on the image, where such artificial artifacts are introduced and hence, we can remove them in the algorithm through our post-processing pipeline.
5. We have purposefully chosen, not to remove these regions in our results so that the raw output from the algorithm is clearly communicated.



Method-1 Results

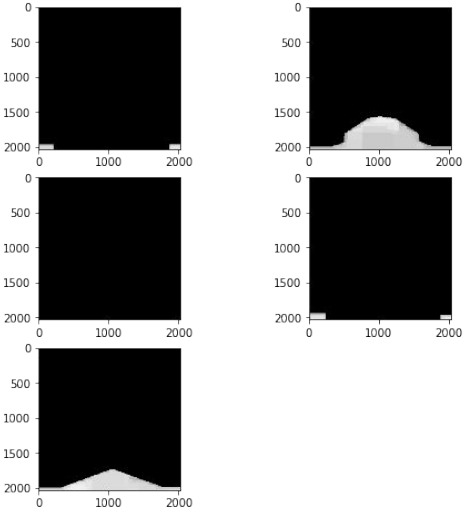


Step -1: Max gradient image

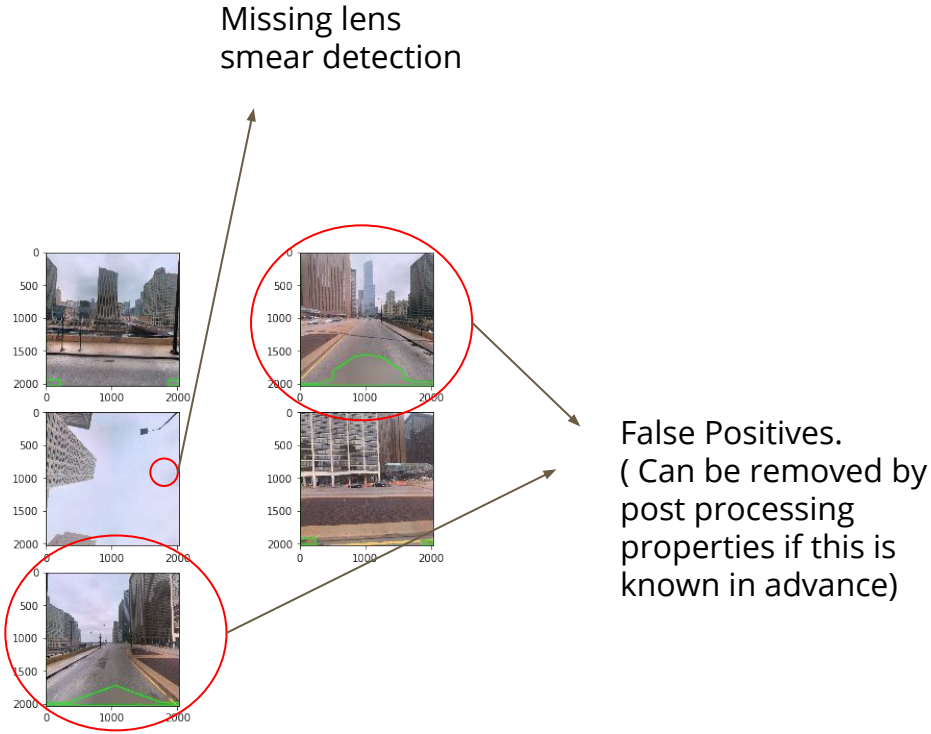


Step -2: Inverse binary thresholded image

Method-1 Results



Step 3: Dilated image



Step 4: Final result

Method -2 Core ideas and observations

1. Another alternative way of defining lens smear is to be smooth but dark regions in an image
2. This definition of smear lends into an easy detection mechanisms but with some false positive cases.

Method-2 Detailed explanation

1. **Image preprocessing:**
    a. Do histogram equalisation to enhance contrast
    b. Split images into small batches.
2. **Core algorithm:-**
    a. Calculate the average image of the batch sequence
    b. Now the image is thresholded to create a binary image.
        i. Global threshold: threshold the image at a low value (The specific value used was 70). The justification behind this value lies in the observation that lens smear always tended to be darker than their surroundings.
        ii. Adaptive threshold: Used a gaussian window to deter
    c. Invert the binary image
3. **Post processing of results:-**
    a. FInd all contours on the binary image and draw these contours on the original image

Method 2- Code explained

1. Three functions: find_mean_img, create_mask, show_smudge
2. Find_mean_img:
    a. Creates a mean image by first adding up the values for each pixel of each image, then dividing the summations by the number of images.
    b. Resized large images to (1000 X 1000) for processing and consistency.
    c. -Created a parameter to change number of photos processed.
3. Find_mask:Used cv2.adaptiveThreshold to binarize the mean image or used a global threshold for binarising the image

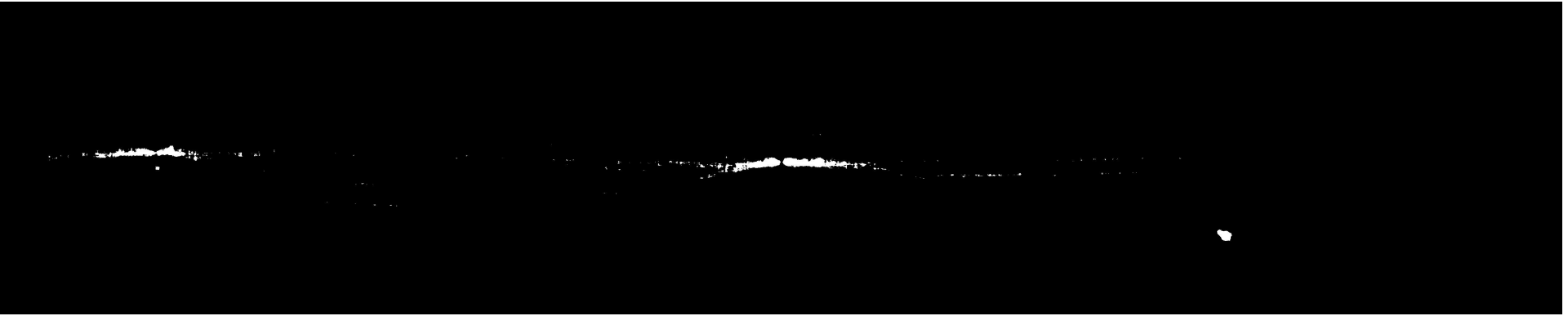
    a. Used an adaptive Gaussian threshold algorithm which creates weighted sums for small localized blocks throughout the image

        i. Created a grayscale of mean image before thresholding
        ii. Two parameters for adaptive Threshold
            - subtracted_constant: changes a general thresholding value for the image
            - thresh_block_size: changes the size of each localized block
4. Show_smudge:
    a. finds the contours in the threshold image with cv2.findCountours
    b. get area for each contour, and check if it is within a max and min size range declared as parameters (A post processing step_
    c. if a smudge is found, it is drawn on a random image with cv2.drawContours

Method-2 Results: Median Images

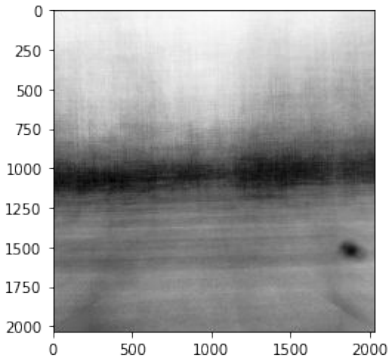
Cam_0 Cam_1 Cam_2 Cam_3 Cam_5

Method-2 Results: Threshold Masks

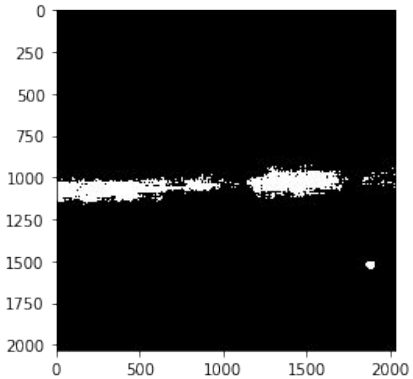


Cam_0 Cam_1 Cam_2 Cam_3 Cam_5

Method 2 - Results - Global threshold with no contour post processing

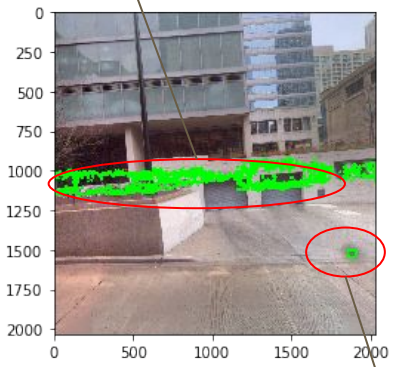


Step 1: Calculate Average image



Step 2: Binarize image

Wrong detections



Step 3: draw contours on image

Correct detections

Method-2 Results: Adaptive thresholding with contour post processing

Cam_3

Method 2 - Downsides

- This method is not ideal because the resulting contours depend heavily on the arbitrary parameters chosen during thresholding and contour size limits.
- Despite averaging several thousand photos, there were still large contours appearing near the center of certain camera's mean photos that were not smears. If there was a large smear on the lens this method would not detect it.
- There are sometimes dark areas in the image, which don't belong to a lens smear
- Through our lens smear detection rate would be good in this case, our false positive counts would be high since a lot of other regions are detected in addition to the actual lens smear

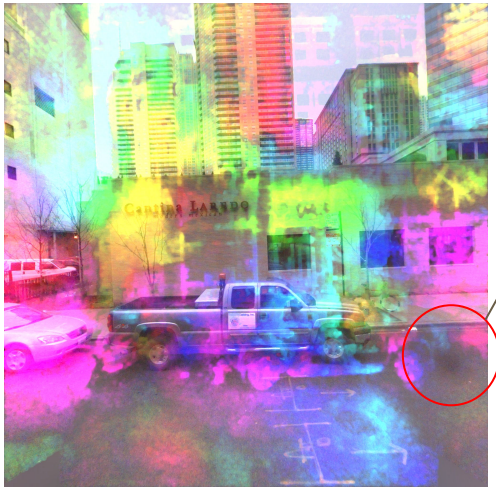
Method -3 - Core ideology and observations

1. Since the vehicle moves, almost all of the pixels of the camera are in constant motion
2. But since the lens smear sticks to a lens, pixels associated with these show nearly zero pixel motion. This property can be exploited to isolate areas where lens smear happens.

Method -3 - Detailed explanation

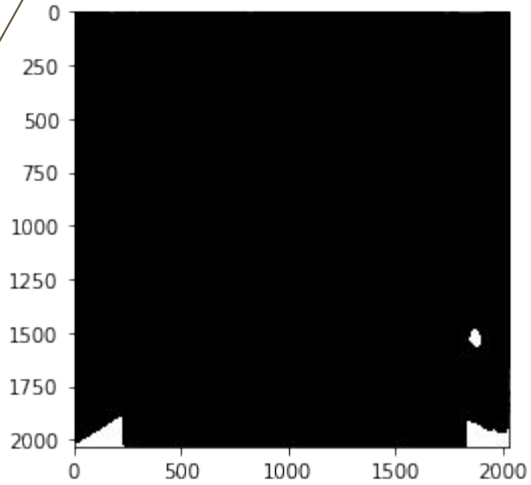
1. **Image preprocessing:** Do histogram equalisation to enhance contrast
2. **Core Algorithm:**
    a. Calculate a dense optical flow of all pixels in the image across a given a sequence
    b. Find all pixels whose optical flow across the sequence was nearly zero (The value hyperparameter in practice turned out to about 20 since sometimes smears disappear and reappear in images)
3. **Post processing:**
    a. Find contours on the mask and display the detected contours on the image

Method 3 - Results

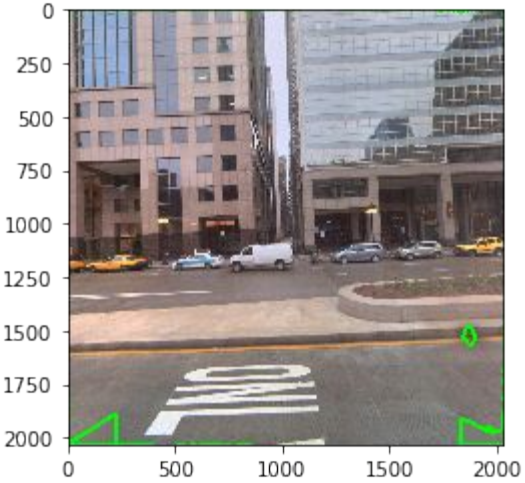


Optical flow estimated for one image in the sequence

Optical flow is zero in a lens smear region

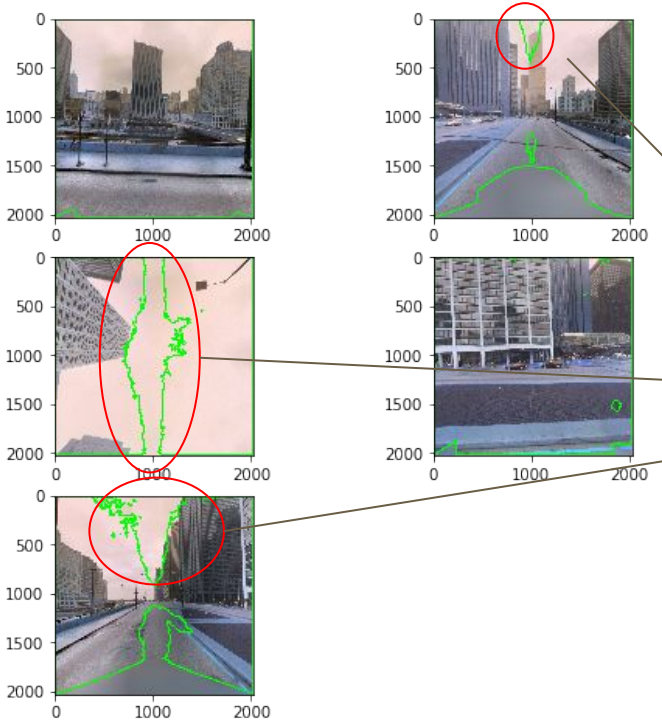


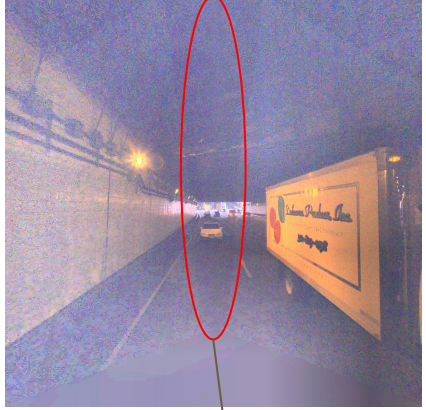
Masked image showing regions where optical flow was very small and hence the lens smear regions

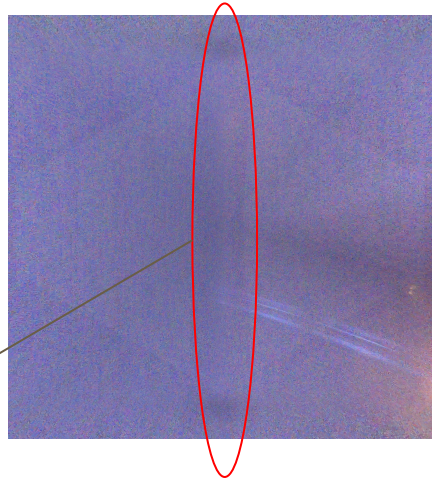


Final result

Method 3 - Results







When we carefully went through the dataset, we found some smear patterns.

But it's hard for us to make a firm conclusion that is a lens smear but we are sure that these are definitely due to imaging problems since these weird patterns are visible in a majority of images across the sequence

Method3 - Competitive Analysis

1. Method3 was by far the best method, since it made very weak assumptions about the data.
2. But it's easy to foresee that it may miss on potential detections if the smear is not visible across the majority of the image. But it must be noted that this method was able to detect smear patterns which were went missing in a few images also
3. The biggest advantage of this method is the presence of only one intuitive of hyperparameter - The max value for pixel movement across the frames. This value can be kept a conservative low value so that this detects most of the smears.
4. Unlike other methods, this method is very robust: It does not make any assumption on the lighting or the specific structure of the smear itself

Other failed strategies

1. **Modelling lens smear as a gaussian distributed color patch:**
    a. Some of these lens smear have a darks colors at their center of the patch and their color degrades as a function of radial distance from the point of origination.
    b. We designed a filter to detect this kind of gaussian color distribution.
        i. For a given window size (kernel), We calculated the mean of that pixel and we also multiplied the same patch with a gaussian kernel. If these two values are the nearly same, the patch of pixel currently being view obeys a gaussian distribution of color.
    c. **Downsides:** The size of the patch may vary and hence, we have to do a brute force search along all feasible window sizes, thus making this approach impractical.

2. **Modelling each pixel histogram across a sequence:**

    a. For each pixel, we calculated its histogram distribution across the whole sequence. The hope was that since the pixels in a lens smear don't vary a lot, we could classify each pixel as belonging to a smear or not belonging to a smear based on the variance of the histogram associated with each pixel.

    b. But, since the smears vanish in a few images, the histograms associated with smear pixels ended being nearly carrying the same variance as a non-smear pixel histogram.

Big data processing - Good and bad strategies

1. **Bad Strategies:-**
    a. **Resize image:** Since the image resolution is too high, one would be tempted to reduce the image size. But the size of lens smear in many of these images is very small and therefore, resizing an image defeats the purpose of doing the exercise.
    b. **Reading all images into RAM:** This is impossible since the data set is very huge to be loaded into RAM. Therefore, it is useful to decompose the code into image wise operations.
2. **Good Strategies:-**
    a. **Numpy vectorisation:**
        i. Iterating through every pixel is costly. As an example, iteration through every pixel and comparing the value in that pixel to the same pixel in another array takes more than 15s on my PC. But the same operation only takes a few ms due to numpy vectorisation
        ii. **Sampling image:** Rather than processing all images, smears for a sequence could be founding by leaving images out at regular frequencies. The vehicle is moving slow enough to enable us to make this downsampling strategy. Datasets with 4000 images gave nearly the same results even after being downsampled to 400 images

What is numpy vectorisation?

1. Numpy vectorisation of a code refers to using inbuilt numpy functions or matrix multiplications, thus avoiding native iterations using Python loops
2. **Why is this useful:**
    a. Numpy vectorised code runs in C and thus faster. Python loops are slower due to fact that loops typically loop through iterables which can contain heterogeneous data types
    b. Numpy code has SIMD (Single Input Multiple Data) instructions written in Assembly. This translates to writing SSE instructions for Intel and NEON instructions for ARM. Thus, numpy has parallelism at its core, thus speeding up computations.

```
sobel = cv2.Sobel(image,cv2.CV_64F,1,0,ksize=5)
max_gradient = np.maximum(sobel, max_gradient)
```

An example of numpy vectorisation. This code takes a few ms but if the same code is written in python native for loops, it takes as high as 15 S.

Conclusion

Some challenges with detecting smears are

1. The images contained other somewhat artificially induced blurring. This often confused with actual lens smear. It would safe to assume that this blurring won't exist in the real time data.
2. Lens smears sometimes disappear with lighting. This makes it difficult to detect them, since the assumption that a lens smear causes somewhat consistent damage to an image across a sequence no longer holds.