Geospatial vision & visualization: HW1

Levi Todes
Jordan Haskel

First attempt:

The first attempt was to perform background subtraction on the images without any pre-processing. The thought behind this was that the smudge would be the same between frames however the rest of the frame would change since the car was moving. This did not yield good results.

The results can be seen on the right.

In each frame the left is the original and the right is the output of background subtraction

The first attempt was to use MOG2 background subtraction:

The second attempt was to use KNN background subtraction:

First attempt improved:

The next attempt was to perform histogram equalization to make the smudge more pronounced in the images.

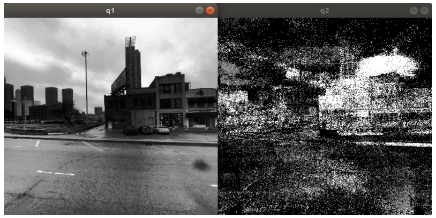
Then background subtraction was tried again with some improvement but not much.

The results can be seen on the right.

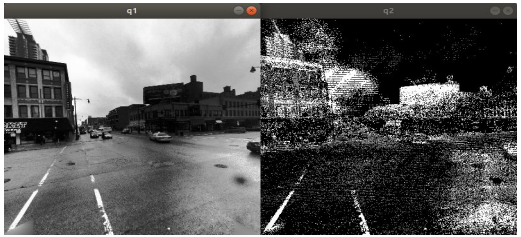
The smudge is definitely more

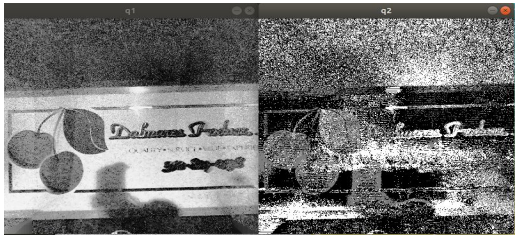
visible here.

The MOG2 background subtraction after histogram equalization:



KNN background subtraction after histogram equalization:



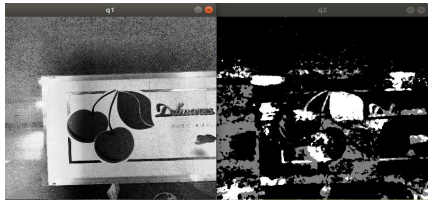


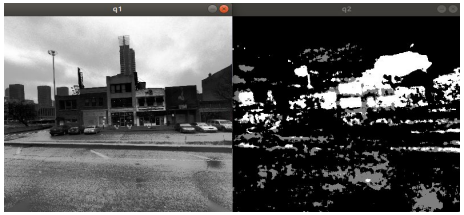
First attempt more:

The new images now were very noisy so median filtering was applied. This fixed the noise but the smudges still were not evident in each frame and the background subtraction was still catching alot of other objects and often missing the smudge.

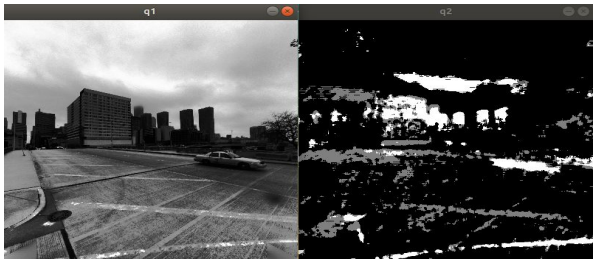
The results can be seen on the right.

The median filtered MOG2 background subtraction after histogram equalization:





The median filtered KNN background subtraction after histogram equalization:



Second attempt:

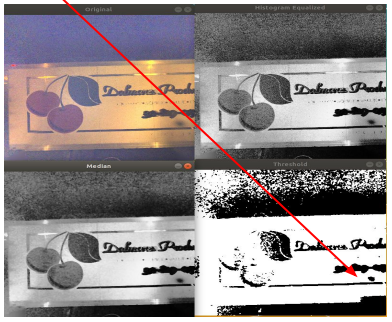
Since attempt 1 wasnt proving useful, attempt 2 was based on the fact that the smudge was generally darker than the surrounding pixels in the image.

Therefore threshold filtering was used to separate pixels based on value.

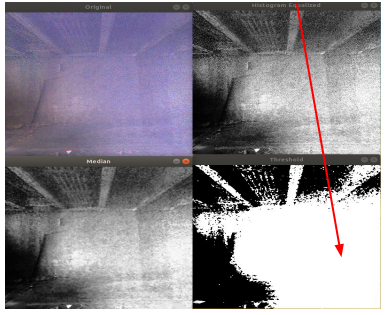
A threshold value of 90 proved to be good.

This was done on the images that had undergone histogram equalization and that had been median blurred.

This worked very well at getting the smudge in most frames



However there were problems based on the brightness of the frame



Second attempt:

The brightness of the frame was first addressed by calculating a threshold based on the mean value of the pixels however this mean did not vary much.

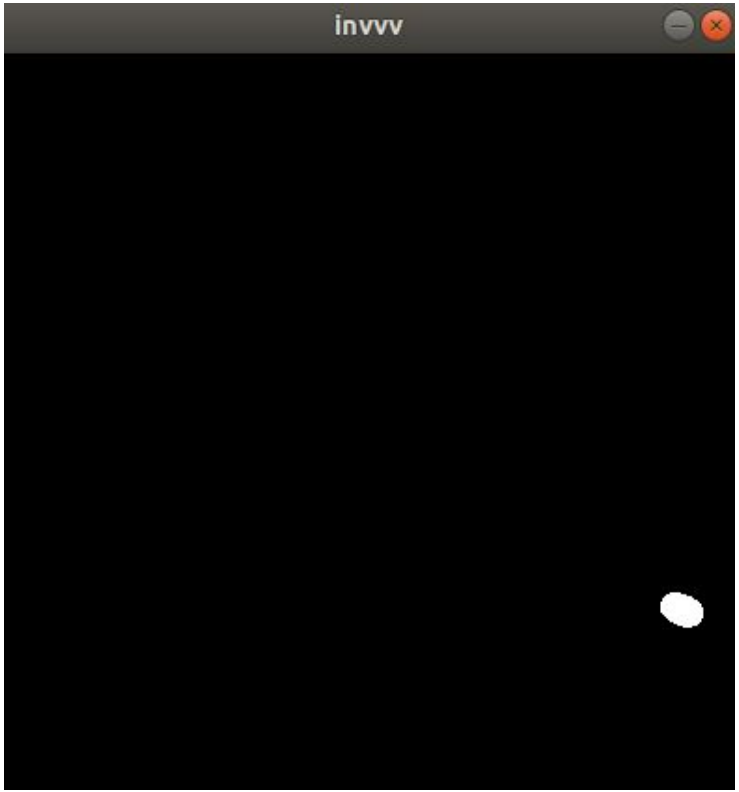
In order to get a good idea of local brightness the frames were split into 4 frames and the threshold values for each was calculated on the mean of each.

Thresh = mean-(mean/5)

Stitched and median filtered

Creating the Mask (effort 1):

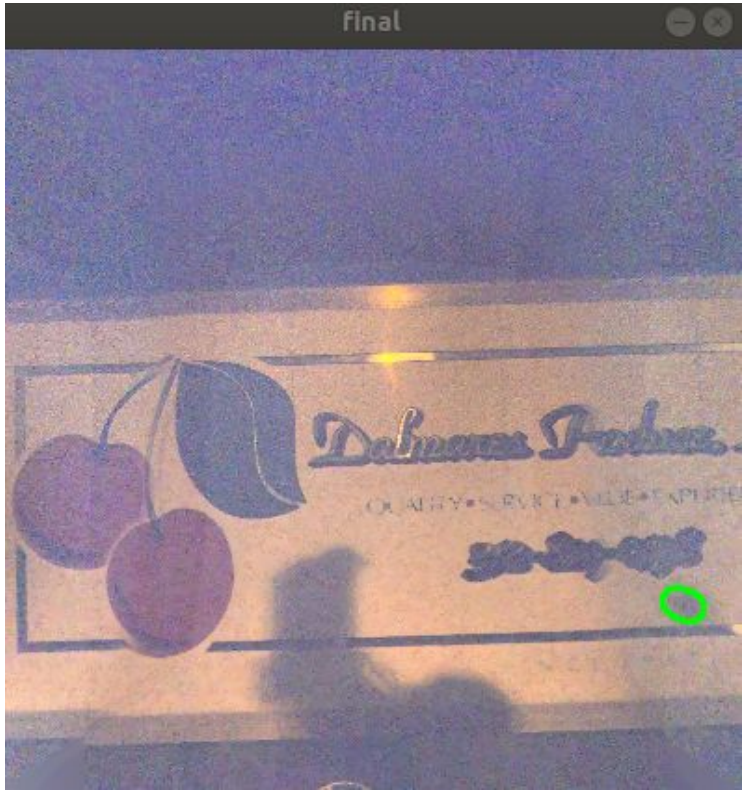
Once the the smudge was easily visible due to the preprocessing described earlier, the approach was to take the average (mean) value of all of the preprocessed frames and then use a threshold on that average so as to create a mask with the average position of the blur over all the images.



Locating smudge (effort 1):

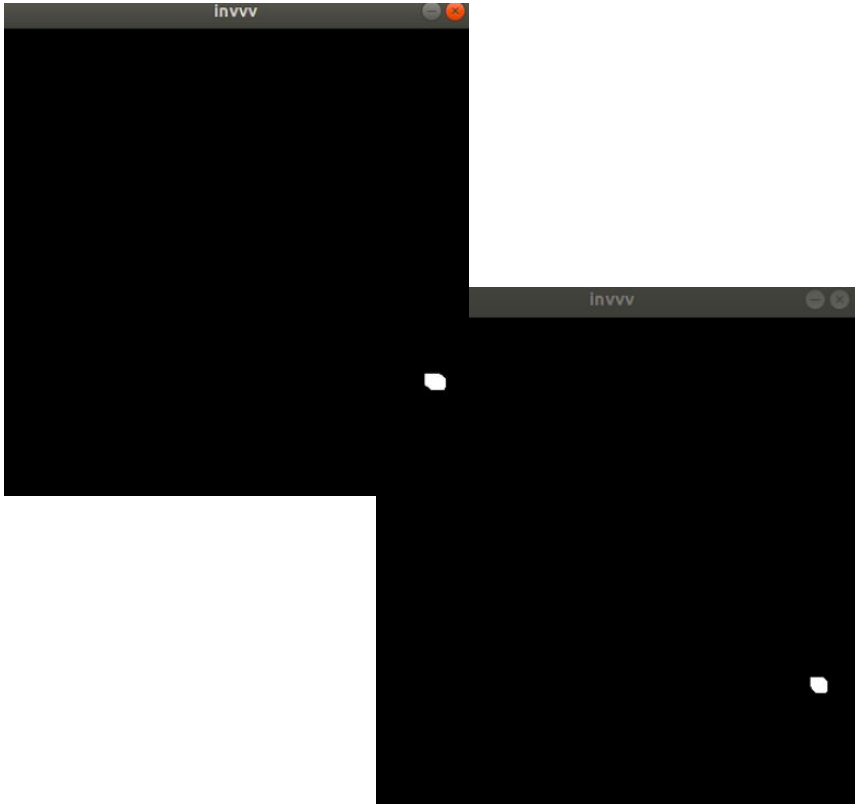
Using the mask created, the contours of the mask image were found and superimposed onto the original frames so as to circle the location of the blur.

This effort was successful, however, it assumes the smudge is in the “average” place every frame. This lead to effort 2.



Creating the Mask (effort 2):

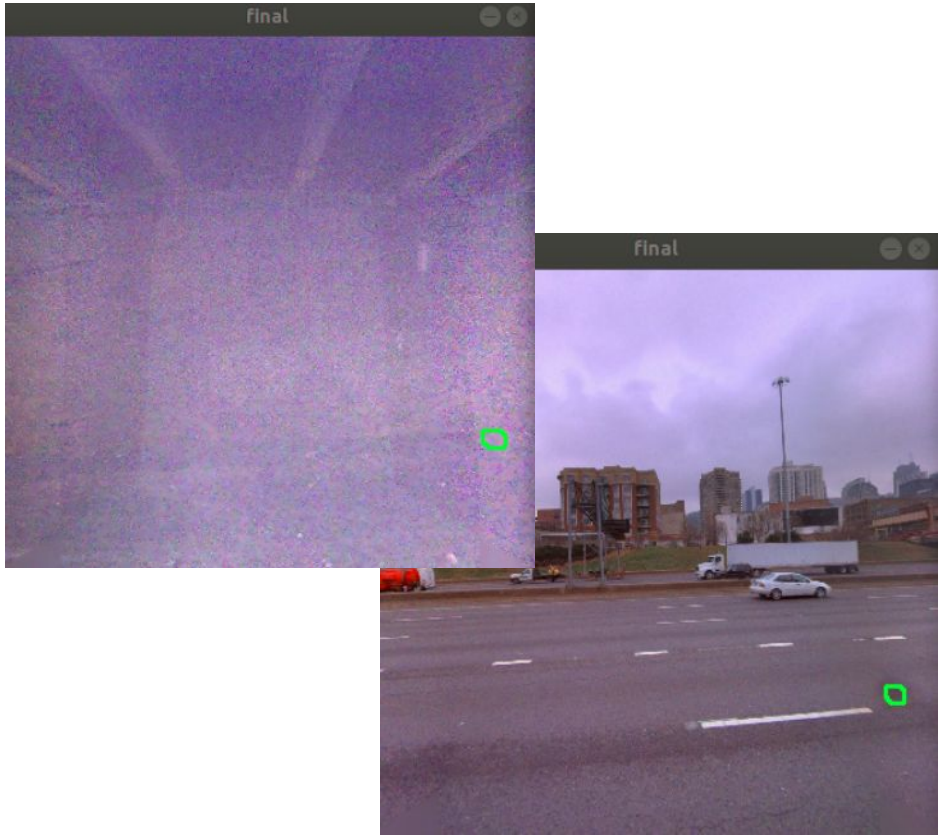
This effort, similarly calculates an average (mean) value on the preprocessed frames but uses a rolling window of frames over which to calculate that average.This is then run through the same threshold as before. The result is an adaptive mask that updates itself every 20 frames. This mask lets a few false positives through the threshold, but these are somewhat solved by using an Erosion and Dilation on the resulting mask.



Locating smudge (effort 2):

Using the mask created, the contours of the mask image were found and superimposed onto the original frames so as to circle the location of the blur.

Using the new adaptive mask, it becomes necessary to set a minimum and maximum area for the contours found and only draw those within that boundary



Final System Block Diagram