Image Enhancement of an Urban Driving Scenario to Improve Object Detection

[redacted]

Date: 22/02/2022

[redacted]

Abstract — To improve the objects detected withing an urban driving dataset, I used a validation image set, to initially check any image enhancement techniques and a test set. I made use of my own image processing Python file to produce corrected images, a YOLO recognisation algorithm, which produced a video output and score for my processed images, and a compare images script, to give the SSIM, mean squared error (MSE) and the mean absolute error (MAE) between my images and the ground-truth images in the validation set. I found the best performance, of both subjective and objective metrics, involved removing noise through a median filter, dewarping the image, and applying a contrast stretch and CLAHE equalisation.

Index Terms — YOLO – 'You Only Look Once' image detection algorithm which recognises objects within a frame, SSIM – 'Structural Similarity Index' which scores the retention of image quality between two images, CLAHE – 'Contrast Limited Adaptive Histogram Equalisation' which is a unique form of global image equalisation, Harris Corner Detector – an accurate algorithm, commonly used in computer vision, to extract the location of corners within an image.

1 Introduction and Related Material

IN this report, I will be exploring the process I underwent to better detect objects. Specifically, objects found within greyscale images from an urban driving scenario. I have assumed that through correcting the images they will perform better with the object detection algorithms.

To improve the images as best as I could, I took into consideration the subjective visual image quality and the objective YOLO scores and the comparing images script scores.

2 Solution Design, Implementation and Results

After analysing the dataset, I split my methods into three sections including noise removal, dewarping and brightness and contrast adjustment, since these all required adjusting.

2.1 Noise Removal

Upon close inspection, the dataset contained a lot of salt and pepper noise and gaussian noise, as seen in Figure 1.

Fig 1. Zoomed-in first image of the test dataset, demonstrating the salt and pepper noise and some gaussian noise.

I first decided to try applying a median filter, since this works best at removing salt and pepper noise, when compated to mean and conservative filtering, and also removes gaussian noise well. Smoothing however is a lossy transform and high frequency detail is lost [1]. Hence, I opted to define three neighbours, finding this to be the best balance between removing noise and reduced smoothing.

I also attempted to use non-local means and a bilateral filter. Although these made the images better resemble the ground truth, they each retained lots of hidden salt and pepper noise before smoothing the image too much (Figure 2).

Fig 2. An image from the test data set, after non-local means smoothing is applied.

Additionally, a gaussian filter would blur too strongly. As a result, since the median filter would perform best in removing all the noise, I applied this to all the images in the test set to produce qualitively better images. This resulted in the image below (Figure 3).

I applied all these filters through OpenCV's inbuilt functions.

Fig 3. An image from the test data set, after median smoothing is applied.

2.2 Dewarping

Next, I decided to dewarp the images, since the camera did not output any images in a standard size but included a strangely shaped border, as seen in Figure 4.

Fig 4. The first image of the test dataset, demonstrating the border around the images.

Since dewarping would correct the aspect of objects, this would reduce the SSIM, increase the MAE and MSE and also improve the performance of YOLO. Without dewarping we get an MAE of ~0.45, which is very low.

To perform such an operation, I implemented my own method. I first blurred each image using a gaussian filter and then applied OpenCV's canny function to create a boolean black and white image - colouring the edges in white. Then I used OpenCV's Harris corner detector, saving the corners to a list. I extracted the most repeated coordinates from this list, which refer to the four corners of the above image. I hard-coded these coordinates to create a perspective transform and apply this to each image. This resulted in images such as in Figure 5.

Fig 5. Dewarping and noise removal applied to the image in Figure 4.

2.3 Brightness and Contrast Adjustment

As seen by Figure 6, both the contrast and the brightness of our image were increased.

Fig 6.1 and Fig 6.2 Colour value differences between the matching corrupted and ground-truth images from the validation set.

To define clear edges and improve the images after smoothing, I attempted to apply a laplacian filter, however this generated a lot of unnecessary noise. Instead, I used a small edge sharpening kernel, but this reduced the visual image quality and maintained the YOLO score of 0.682…, and so I removed this also.

Fig 7.1 and Fig 7.2 YOLO final frame output, with and without sharpened edges, respectively.

To restore some of the overexposed areas, I tried applying a power law transform, however this also led to more noise. Instead, I applied a contrast stretch, manually defining the lower and upper bounds of the colours, followed by CLAHE with a smaller grid-size - through trail and error, this nicely defined further areas of the images.

Fig 8.1 and Fig 8.2 An image, from the validation set, adjusted up to this point, and the matching ground-truth image

Finally, to slightly reduce the brightness, I applied a gamma correction algorithm. However, this reduced the YOLO score to 0.678…, and so I decided to remove it, even though it improved the visual look of the images. I could have applied a Butterworth low-pass filter [1], however ringing was not visually present in the produced images, so I did not choose to include it.

3 Conclusions and Further Comments

The order in which I performed my operations did affect YOLO very much, since they acted independently of each other.

The final YOLO score achieved on the test set through my process was 0.682… When applying my process to the validation set and comparing these to the ground-truth, I achieved an SSIM of 0.762…, MSE of 0.0444… and MAE of 0.163, which all appear to be reasonable results.

Although these values are ~0.1-0.2 higher than when edge correction and brightness adjustments are made, I believe I found the best balance between visually good images, detecting objects and generating images close to the original, which I sought out to do.

References

[1] Atapour-Abarghouei, A. (2021). Image Processing Lecture PowerPoints 1 – 10. *Data Science Lectures - COMP2271*, Durham University, delivered 4 October 2021