Processing poor-quality images to improve object-detection performance

Module Name: COMP2271
Date: 27/01/2022
Submitted as part of the degree of BSc Natural Sciences to the
Board of Examiners in the Department of Computer Sciences, Durham University

1 Introduction

We are given two sets of grayscale images: a validation set and a test set, of a vehicle driving through Durham, United Kingdom. Both sets include artificially corrupt versions of the images, while we are also given the original, uncorrupted images for the validation set. The corrupt versions contain substantial amounts of noise, are warped, and suffer from poor contrast and brightness. Our aim is to use image processing techniques (using OpenCV and Python) in order to revert the corrupt images to their original state, as best as possible, to improve the performance of object-detection methods.

Fig. 1. An image from the corrupt validation set, and the corresponding image from the original, uncorrupted validation set. Note all of the corrupt images have been artificially modified in exactly the same way, so we can consider processing techniques on only one of the images.

2 Processing the images

2.1 Noise removal

We see in Fig. 1 that the corrupt images contain a lot of salt and pepper noise, and a lot of Gaussian noise. It makes sense to get rid of the salt and pepper noise first, which we can do easily by applying a 3×3 median filter onto the images. Afterwards, we can remove some of the Gaussian noise by applying a 3×3 non-local means filter to the images.

Fig. 2. The image shown in Fig. 1 after we have applied noise removal techniques on it. Notice most of the noise is gone, particularly the salt and pepper noise. The image is now slightly blurrier than before as a result of the noise removal, but we will fix this later.

2.2 Dewarping

The images are warped as shown in Fig. 1, but thankfully they are not warped so badly, so dewarping them is easy using methods in OpenCV. We simply find the coordinates of the "true" corners in the warped images, which can be done using NumPy, and then use OpenCV to essentially map those coordinates to the actual corners of the image, using a perspective transform.

Fig. 3. The image shown in Fig. 2 after we have dewarped it.

2.3 Brightness and contrast enhancement

The corrupt images also suffer from poor contrast and brightness. A usually efficient technique to solve this is

histogram equalization, so we apply this to the images and observe the results. However, despite this method performing well in some areas of the image, a lot of information is clearly lost, and more noise has been added, as we see in Fig. 4.

Fig. 4. The image shown in Fig. 3 after applying histogram equalization on it. It does a good job of obtaining a similar shade of gray for parts of the road compared to the original, but otherwise we see a lot of information has been lost and there is added noise, so we must try other methods.

One thing we notice about the corrupt images is that there are regions which are very bright and have little visible detail, while there are also very dark regions. To solve this problem, we can apply a logarithmic transform, followed by an exponential transform (with basis 1.0175), with a small amount of γ-correction afterwards (with $\gamma = 0.95$). In general, this will widen the dynamic range of darker regions, and tighten the dynamic range of brighter regions.

$$\begin{aligned} p_{output1} &= c_1 \times \ln(1 + p_{input}) \\ p_{output2} &= c_2 \times (1.0175^{p_{output1}} - 1) \\ p_{output3} &= (p_{output2})^{0.95}, \end{aligned} \tag{1}$$

where c_1, c_2 are scaling constants.

Fig. 5. The image shown in Fig. 3 after applying the point transform steps in (1). We have been able to increase the brightness and improve the contrast somewhat, but it is very difficult to get good results with these images, as a lot of the pixels are corrupt in such a way that their value is 255 (white), so much of their information and that of the surrounding areas has been lost

2.4 Sharpening

As a side effect of all the processing techniques we have applied, the images have become slightly blurred. To rectify this, we can sharpen (de-blur) the images, by convolving them with a standard 3×3 sharpening kernel [1].

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Using the convolve2d() function with the above matrix from SciPy's signal package, we are able to sharpen the images, returning images which are less blurred and closer to the original images.

Fig. 6. The image shown in Fig. 5 after we have sharpened it.

3 Evaluation

We compared our processed images from the validation set with the ground truth, and obtained three different scores evaluating the error between the two sets, and how similar they were. From Table 1, we see that our processed image set was around 80% similar to the original validation set, and that the errors were fairly low. By comparison, the corrupt images were only around 37% similar to the ground truth.

TABLE 1
Performance metrics results of processed images against ground truth from validation set (to 3 significant figures)

Metric	Score
Structural Similarity (SSIM)	0.797
Mean Squared Error (MSE)	0.0209
Mean Absolute Error (MAE)	0.107

As well as processing the corrupt validation set, we processed the corrupt test set in the same way.

Fig. 7. An image from the corrupt test set, along with the corresponding processed image.

Although we don't have a ground truth set for the test images, we can see that qualitatively, the images have

improved due to our processing techniques, and show more detail in many areas.

After viewing the processed images, we then created a chronological video from the image set, at 3 frames per second (FPS), and ran the video against the YOLO object detection algorithm, which outputs a score from $0-1$ based on its performance. When we ran YOLO with the corrupt test set, the algorithm output a score of around 0.66. By contrast, our processed test set was able to achieve a score of 0.7, showing some quantitative improvement in our new image set.

4 Conclusion

In conclusion, we were able to somewhat improve the quality of the images in both sets via the processing techniques discussed in this report, however the corrupt images were of very poor quality in some aspects, specifically contrast and brightness. For that reason, it was hard to obtain processed images that clearly stood out from their corresponding corrupt ones, both in terms of quantitative measures, and subjective human assessment.

References

[1] https://en.wikipedia.org/wiki/Kernel_(image_processing)