Image Processing Report

Alex Dinan, vwzn63, (679 words)

1 Image Inpainting

Firstly, I implemented a state-of-the-art, exemplar-based inpainting algorithm as proposed in [1]. Unlike for diffusion-based methods, appropriate patches are copied from the source region, resulting in minimal image smoothing. Additionally, a greedy best-first filling strategy, processes patches on the fill-front in priority order. Patches with high confidence, such as those surrounded by reliable pixels, as well as patches on the continuation of strong image edges are processed first. This ensures correct propagation of linear structures and robustness to the shape of the filling region.

However, one significant drawback I encountered was computational overhead — particularly in finding the most similar source patch to a given target. To address this, I considered patches row-by-row, skipping computations unlikely to yield an optimal solution according to the following rules.

$$d(\Psi_{current}, \Psi_{target}) \geq n \cdot d(\Psi_{best}, \Psi_{target}) \implies \text{skip next m patches in current row} \tag{1}$$

$$d(\Psi_{current\_row\_best}, \Psi_{target}) \geq \frac{3}{2} n \cdot d(\Psi_{best}, \Psi_{target}) \implies \text{skip next m rows of patches} \tag{2}$$

where

$d(\Psi_a, \Psi_b) \rightarrow$ Sum of squared pixel differences (SSD) between patches a and b

$n \in \mathbb{R}_{\geq 1.5} \rightarrow$ opt_factor (algorithm parameter)

$m \in \mathbb{N} \rightarrow$ patch_size (algorithm parameter)

Moreover, by applying inpainting prior to perspective correction, I narrowed the search space for viable source patches, and was able to skip significant proportions of the black background region without consideration.

As demonstrated by Fig.1, these optimisations increased efficiency by up to 3000%, whilst limiting the average total SSD between source and target patches to a rise of 120%. This facilitated the use of a smaller patch size to capture finer image details and textures. An optimisation factor of 2.25, combined with a patch size of 12 provided the best trade off between processing time and image quality, with inpainting taking an average of $4.6s$ per image.

Finally, Fig.2-3 demonstrate the superior performance of the chosen algorithm compared with OpenCV's inbuilt methods which produce significant blurring.

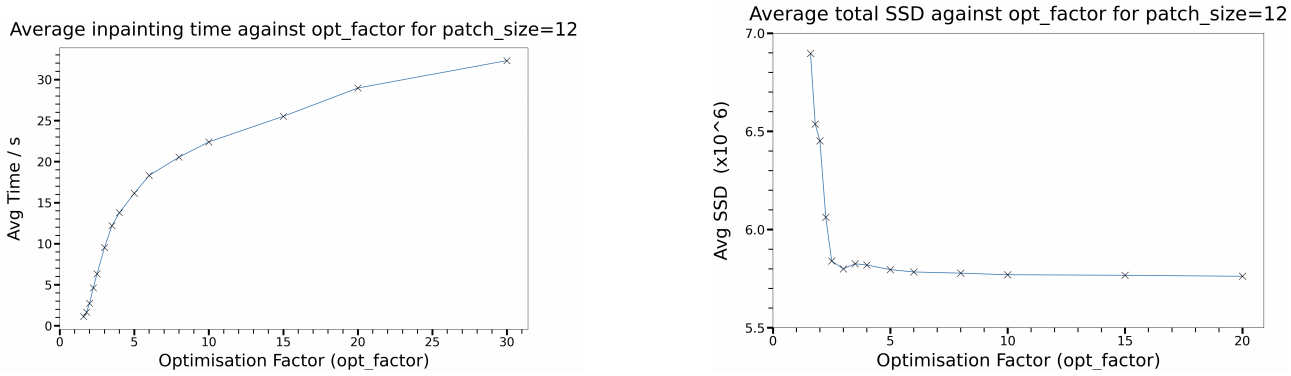


Figure 1: Success of various levels of optimisation in decreasing processing time whilst preserving inpainting quality.

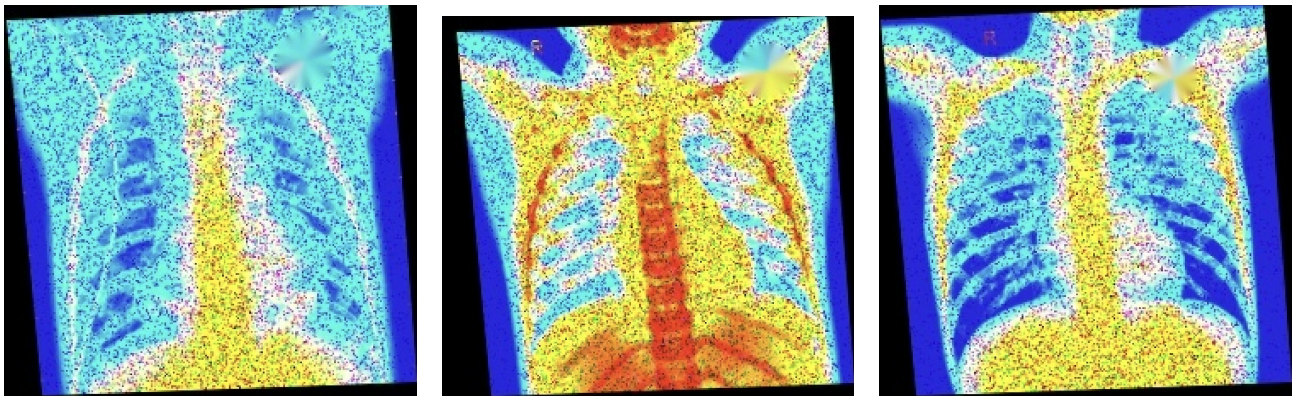


Figure 2: Inpainting results using inbuilt, diffusion-based OpenCV methods

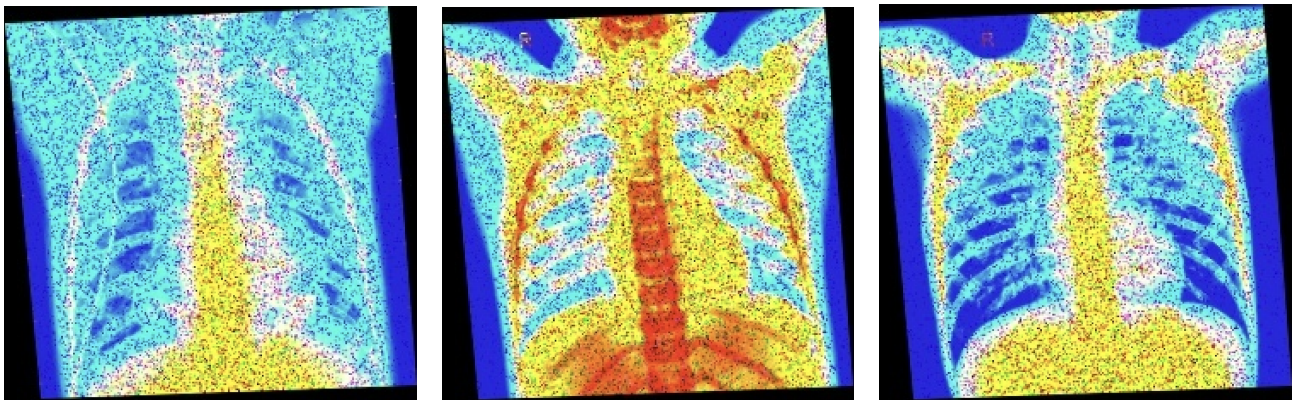
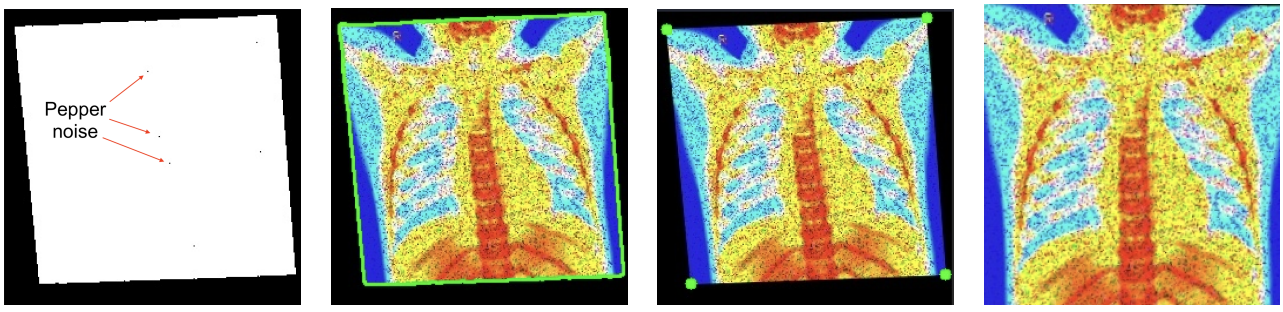


Figure 3: Inpainting results using chosen, exemplar-based method

2 Fixing Perspective

In order to dynamically detect the black background region within the images, I utilised thresholding which produced a binary output. By leveraging OpenCV's contour detection functionalities [2], I selected the contour with the largest area to isolate the background from any pepper noise which may have persisted. Finally, I applied a perspective transformation, mapping the coordinates of the contour corners to the image boundaries.

This technique is significantly more robust than manually hardcoding the transformation coordinates, remaining effective for variations in resolution and perspective warping.



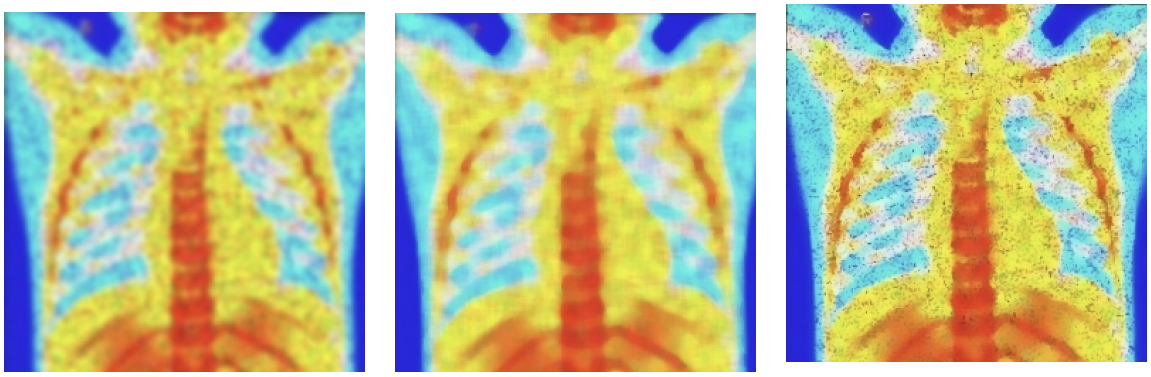
a.) Thresholded image b.) Largest contour c.) Corners of contour d.) After transformation

Figure 4: Visualisation of image dewarping process

3 Noise Removal

Considering the high levels of both Guassian and salt and pepper noise present, local methods such as guassian, median and bilateral filtering [3] were unsuitable (Fig.4). Thus, I opted to use OpenCV's non-local means denoising [4]. This substitutes pixels with a mean of those lying in comparable neighbourhoods across the entire image, weighted by similarity [5]. By considering global pixel data, this approach offers superior noise reduction whilst preserving image details.

Parameters including filter strength, template window size and search window size, were adjusted to optimisise performance of the downstream classifier and strike a balance between noise elimination and blurring.



a.) Guassian filter
Fails to remove Guassian noise

b.) Median filter
Causes image posterisation

c.) Bilateral filtering
Fails to remove salt and pepper nosie

Figure 5: Results of candidate denoising techniques

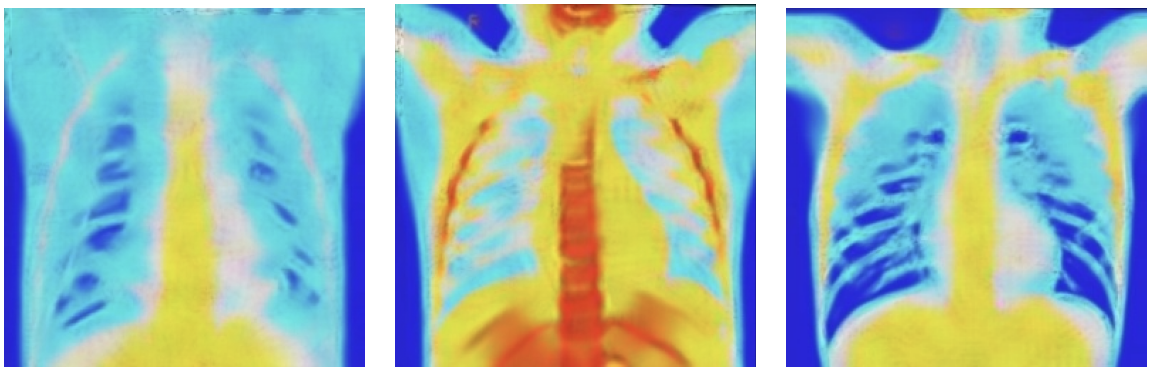


Figure 6: Images after Non-Local Means Denoising

4 Contrast enhancement

To improve contrast, I converted the image to the LAB colour space, separating colour and brightness information. By examining the histogram for the luminance channel, I discovered that the left-most/lowest-intensity peak corresponded to the image background. Therefore by applying gamma correction with $\gamma > 1$, high-value intensities would be spread over a larger range, enhancing image details, whereas the background information would be compressed. This effect is evidenced by the increased height of the background peak in Fig.7, from a frequency of approximately 1200 to 1700.

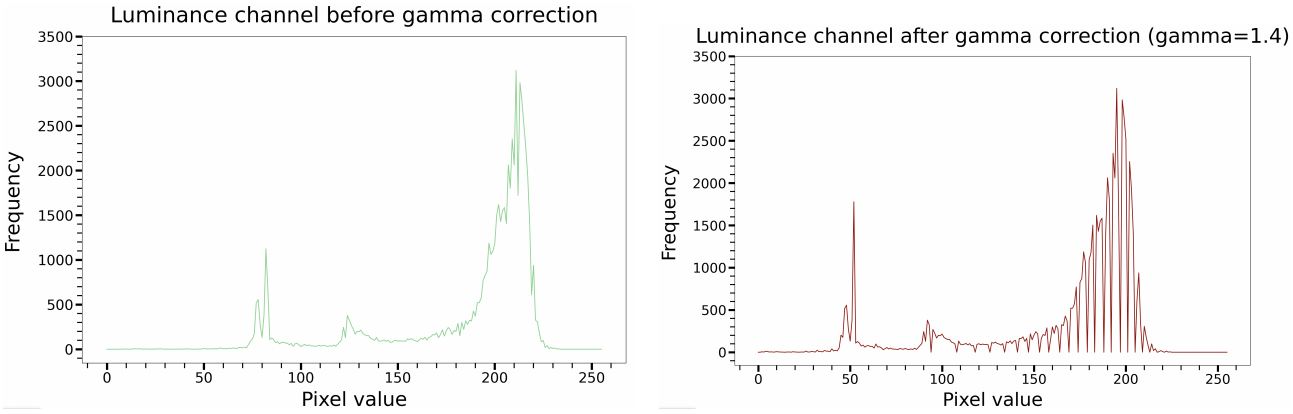


Figure 7: Effect of gamma correction on the luminance channel histogram for a typical image (im060-pneumonia.jpg).

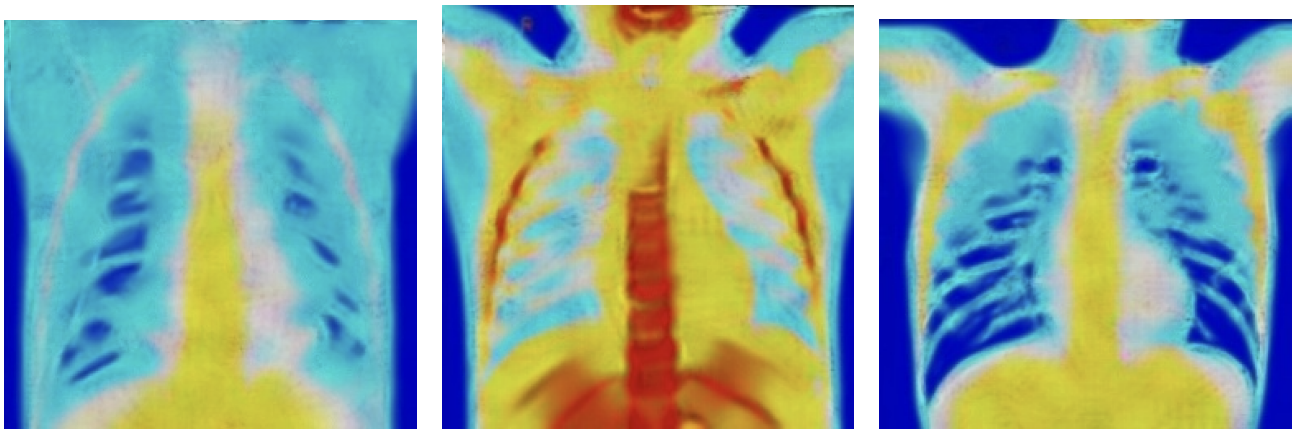


Figure 8: Images after gamma correction ($\gamma = 1.4$)

5 Overall system success

In terms of improving visual quality, the system is successful. The missing region has been realistically inpainted, the warping addressed and noise removed. However, further improvements may arise from reducing the blurring associated with the denoising process and adjusting the colour balance. Specifically, increasing the intensity of green hues to align with the original images.

In terms of performance of the downstream classifier, the system is highly successful, achieving an accuracy of 0.96.

Processing step	**Performance after application**
Initial Images	0.55
Inpainted	0.57
Dewarped	0.81
Denoised	0.93
Gamma corrected	0.96

Table 1: Performance on classifier

References

[1] P. Perez A. Criminisi and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *In: IEEE Transactions on Image Processing*, vol. 13:1200–1212, 2004.

[2] Contours: Getting started. `https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html`.

[3] Image smoothing. `https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html`.

[4] Image denoising. `https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html`.

[5] Bartomeu Coll and Jean-Michel Morel. Non-local means denoising. *Image Processing Online*, 1:5, 09 2011.