# Image Processing Report

## 1 - Introduction

The images were processed as follows:

1. perspective warping
2. inpainting
3. noise filtering
4. colour and contrast adjustment

This order was best, since each step improves those after. For example, the dark background can cause median filtering to remove pixels on the edges of the x-ray. The final results were classified with 0.92 accuracy and have reasonable visual quality. Noise was successfully reduced, preserving most detail, and inpainting plausibly and realistically filled the missing region. Contrast and colour adjustment successfully reveals more details without sacrificing accuracy.

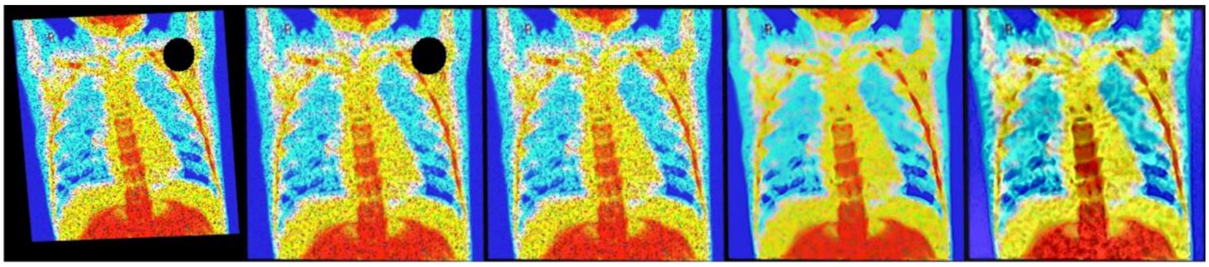


**Figure 1 - image processing pipeline. From left to right: original image, perspective warping, inpainting, noise filtering, colour and contrast adjustment**

## 2 - Image Processing

### 2.1 – It's all a matter of perspective…

I created a mask of the edges of the region of interest (ROI - the x-ray) with a Laplacian kernel and detected its corners using OpenCV's inbuilt method. I then obtained the transform matrix that maps the ROI corners to those of the whole frame, and warped the perspective so that the x-ray filled the frame. I found that leaving a 4-pixel-wide black border increased the classifier accuracy by 0.13 compared to leaving none, without impairing image quality.

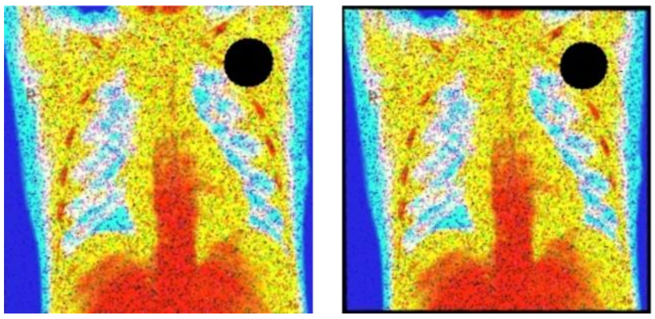
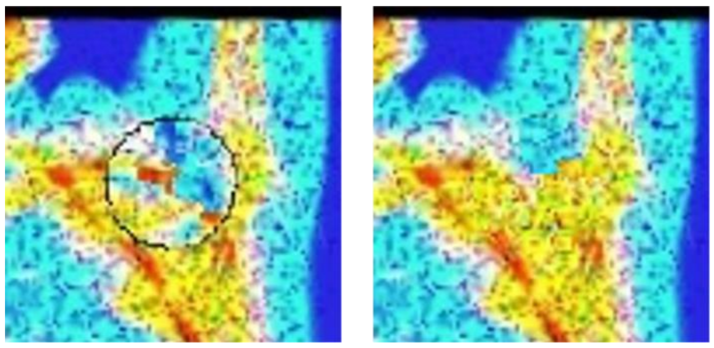


Figure 2 - from left to right: no border around x-ray (accuracy = 0.74), 4-pixel border (accuracy = 0.87)

## 2.2 – Inpainting the missing region

To fill the missing region, I used Criminisi et al.'s inpainting algorithm [1] (specifically, Moura's [2] efficient implementation of it). A mask of the missing region was obtained via thresholding. I dilated it to overlap the missing region with the known region for seamless inpainting.

Figure 3 - from left to right: inpainting using a mask that has not been dilated, using a dilated mask



The algorithm calculates the edge of the mask (the fill front) and gives every pixel ***p*** on it a 'priority' **P(*p*)**, where **P(*p*)** is based on ***p***'s *patch* - a 9x9 neighbourhood centred at ***p***. **P(*p*)** is high if: 1) the patch contains reliable information, meaning pixels in the patch have a high likelihood of being filled in correctly (confidence **C(*p*)**), 2) the patch is on the continuation of a strong edge, calculated using the image's directional gradient (stored as ***p***'s 'data' **D(*p*)**). Then **P(*p*) = C(*p*) * D(*p*)**. Initially *all* pixels have **D(*p*)** = 0, and **C(*p*)** = 0 for pixels in the mask, and 1 otherwise. The pixel with greatest priority is selected as the target pixel $\boldsymbol{p_t}$. All possible patches are compared against $\boldsymbol{p_t}$'s using the sum of squared (Euclidean) distances (SSD) of the pixels already filled in each patch. This step is the most time-consuming. CIELAB is used for its perceptual uniformity, so Euclidean distance more closely corresponds with perceptual difference. The patch with the smallest SSD is used to fill the pixels $\boldsymbol{p_t}$'s patch that are part of the missing region. The fill front and priorities of the pixels on it are recalculated, and this process is repeated until the region is completely filled.

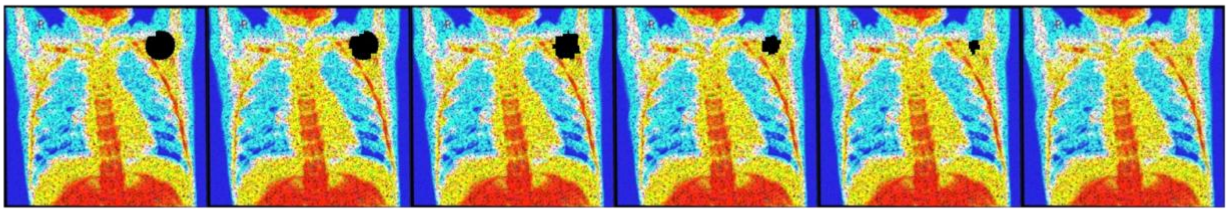


Figure 4 - iterations of Criminisi et al.'s algorithm. From left to right, number of iterations = 1, 4, 10, 30, 40, 48

Compared to OpenCV’s inbuilt inpainting, this reduced classifier accuracy by up to 0.03 but the vast improvement upon visual quality made it a sensible trade-off.

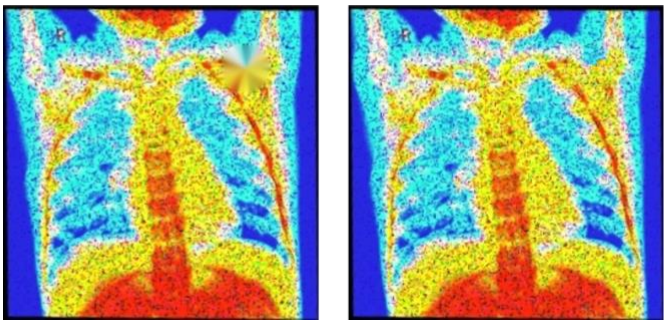


Figure 5 - OpenCV's Telea inpainting (left) leaves significant blurring, whereas the described algorithm (right) fills the region realistically.

The denoising process has three steps: 1) median filtering, 2) non-local means (NLM) filtering and 3) Laplacian edge sharpening.

Median blurring was best at removing salt and pepper (S&P) noise and NLM was effective at reducing gaussian noise. Both blur the image, with more blurring generally resulting in better denoising. So, parameter tuning was vital to strike the right balance. A 3x3 median filter removed all S&P noise with minimal, although obvious, blurring.

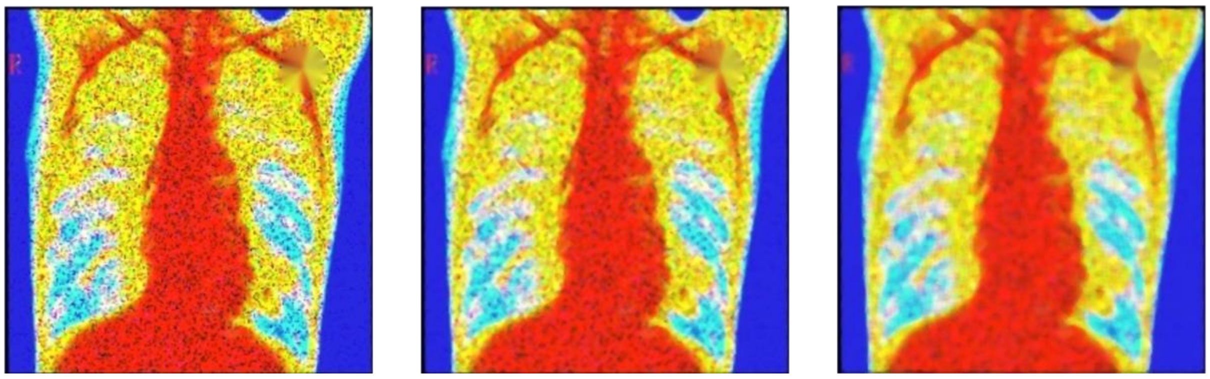
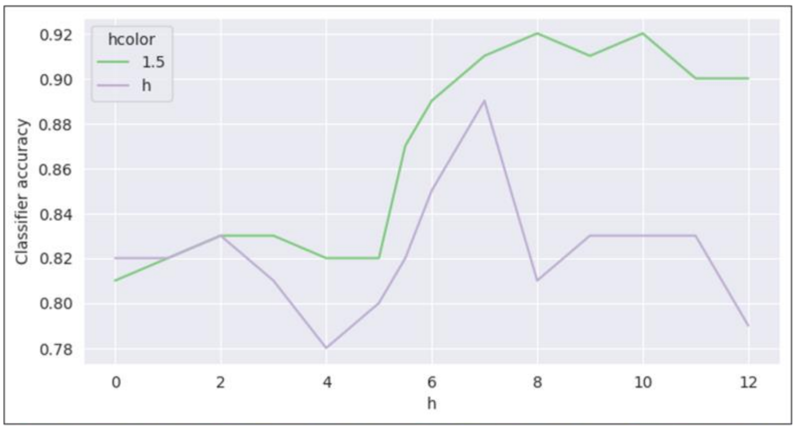


Figure 6 - effect of median filters after perspective warp and OpenCV's Telea inpainting. From left to right: no filter (0.84 accuracy), 3x3 filter (0.81), 5x5 filter (0.81)

For NLM I used OpenCV’s method that uses the CIELAB colour space. Most noise was in the L channel, so a stronger filter strength was used in that than in the A and B channels, reducing the blurring between colours.



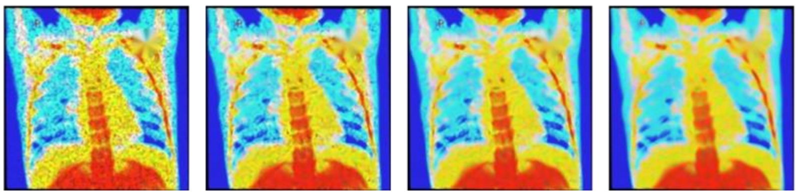


Figure 7 - graph showing change in accuracy caused by varying strength of NLM filter (h) on luminance channel of CIELAB image. The graph also compares the effect of keeping hcolor constant at 1.5 or equal to h, where hcolor is the NLM filter strength in the colour (a and b) components.

Images beneath the graph show the effect of NLM filtering for different h values. From left to right, h = 0, 5, 7, 11.

(Note that NLM has been applied after a 3x3 median kernel to remove salt and pepper noise)

Laplacian edge sharpening reduces blurring but amplifies noise, so it was best to do *after* denoising. I used the kernel: $\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ since others amplified leftover noise too much.

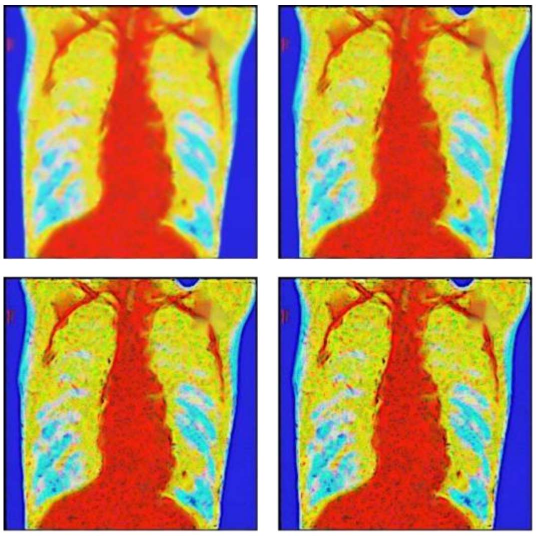


Figure 8 - effect of Laplacian edge sharpening after warping perspective, OpenCV's Telea inpainting, median and NLM filtering. Top left: no edge sharpening (0.9 accuracy); top right: 2D Laplacian kernel using Chebyshev distance as described above (0.94); bottom left: 2D Laplacian kernel using Manhattan distance between pixels (0.93); bottom right: 3x3 Laplacian kernel using OpenCV’s method (0.95).

## 2.4 - Contrast and Colour

I used the CIELAB space for its perceptual uniformity, which prevented contrast enhancement looking unnatural. I used CLAHE in the luminance component to improve contrast and gamma correction to maintain visual quality. This amplified leftover noise, but much more colour and detail became visible. For medical applications where details are vital, this seemed a sensible trade-off.

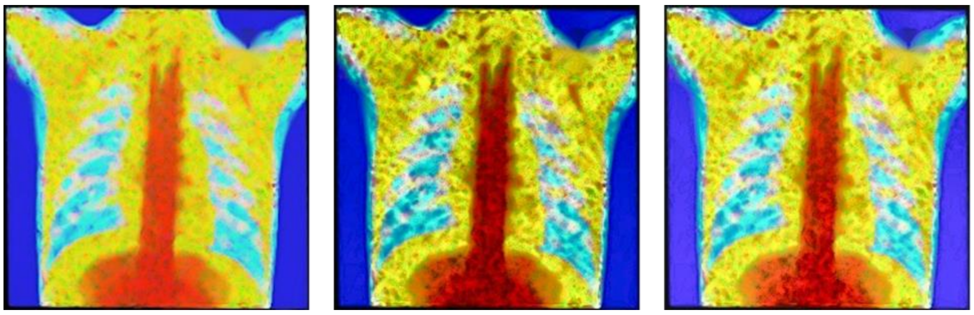


**Figure 9 - From left to right: no CLAHE or gamma correction (0.94 accuracy), CLAHE but no gamma correction (0.93), both CLAHE and gamma correction (0.95). (Note this is after all previous steps where OpenCV's Telea inpainting is used.)**

## Bibliography

[1] A. Criminisi, P. Pérez and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," IEEE Transactions on Image Processing, vol. 13, no. 9, pp. 1200-1212, 2004.

[2] I. C. Moura, "inpaint-object-remover," 2021. [Online]. Available: https://github.com/igorcmoura/inpaint-object-remover. Commit: dc535f2.