## Conclusion

From the analysis it is clear that the parallel version of this particular image processing project is much more efficient than that of the serial version. This is due to the parallel version iterating through multiple pixels at the same time as opposed to one pixel at a time.

The implementation of the 2 parallel versions (accessing the 2d array row-wise and accessing the 2d array column-wise) has no significant difference in the performance. It is concluded that this is due to the fact that the 2D array stores pixel pointers and not the actual pixel value, thus not having any effect on switching between reading from main memory and reading from cache.

The efficiency of the parallel version has been thoroughly checked over and over again to ensure that no thread will be waiting for instructions while another is busy.

The 1st problem encountered was finding a means of inputting an image in terms of its pixel values. This was quickly resolved by the EasyBMP library.

A very interesting problem was encountered when multiplying the kernel matrix to pixels on the edges of the 2D array. It was then resolved by means of edge extrapolation, which ensures that when using the pixels at the edges that there are still values outside the 2D array for computation.

One of the main issues encountered was deciding which kernel matrix would output the best de-noised image, i.e. changing the values as well as the dimensions of the kernel matrix to create the best output image. In that case we have devised multiple examples illustrating the effects of different kernel matrices on a noisy image. We have concluded that every picture (having its own unique noise) will need a different kernel matrix for optimum de-noising.

As a whole, this project does what it was designed to do and that is to input a noisy image and change each pixel of that image by means of accurate calculations to output an image more friendly to our eyes.