Advanced Programming 2023/24 – Assessment 3 – Group Project

Image, Filters, Projections and Slices

Group Name:	**Sort**	
Student Name	**GitHub username**	**Tasks worked on**
Laurel Kyte-Buxton	edsml-lwk16	EdgeDetection.h/ .cpp Merging 2D User interface Report
Yifei (Allison) Dou	acse-ad2123	Volume.h /.cpp ThreeDfilter.h/ .cpp TwoDFilter.h /.cpp Optimise & Performance analysis
Hung-Hsuan (Hanson) Shen	edsml-hs1623	EdgeDetection.h /.cpp Merging 2D User interface Documentation
Xueling Gao	acse-xg1123	ColourCorrection.h /.cpp Documentation and integration of codes 3D tests
Saffron Taylor	edsml-st2923	Image.h/.cpp Filter.h Projection.h/.cpp Slice.h/.cpp Volume.h/.cpp User_3D.h/.cpp Report
Yue Wang	acse-yw3523	2D tests User interface of unit tests Code refactoring Report

1 Algorithms Explanation

1.1 2D Image Filters

1.1.1 Grayscale:

The grayscale filter converts a multi-channel image to shades of grey, reflecting the luminance of the original colours. A weighted sum of red, green, and blue values of each pixel is calculated; it assigns weights of 21.26% to red, 71.52% to green, and 7.22% to blue via the luminosity method. This weighting reflects the human eye's sensitivity, producing a more perceptually accurate grayscale image.

1.1.2 Brightness:

This filter adjusts image brightness by adding a specified value to each pixel's colour channels (R,G,B).The adjustment can either increase or decrease the pixel values, depending on whether the parameter is positive or negative, respectively. This method ensures that the final pixel values remain within the valid range of 0 to 255 through clamping.

1.1.3 Histogram equalisation:

To improve image contrast, this filter redistributes pixel values across the entire intensity range, specifically targeting the luminance aspect in the HSL or HSV colour spaces while preserving hue and saturation. This technique ensures contrast enhancement without altering the original colour information. Initially, images in RGB format are converted to HSV or HSL to isolate luminance ('Value' in HSV, 'Lightness' in HSL). A histogram of luminance values is calculated, and a Cumulative Distribution Function (CDF) is made to remap these values, spreading them across the available range. Finally, we convert the adjusted luminance back to RGB, producing an image with enhanced contrast while maintaining its original colour integrity.

1.1.4 Thresholding:

Thresholding is applied by comparing each pixel's value (in grayscale or a specific channel of HSL/HSV colour space) against a defined parameter (threshold value). Pixels with values above the threshold are set to white (255), and those below are set to black (0), resulting in a binary image.

1.1.5 Salt and pepper noise:

The implementation of salt and pepper noise randomly alters a specified percentage of pixels in the image to be either fully white (salt) or fully black (pepper). This effect is achieved by iterating over each pixel and based on a randomly generated number, deciding whether that pixel should be changed. If a pixel is selected for noise addition, it has an equal chance of becoming black or white. This simplifies analysis by isolating objects or features of interest.

1.1.6 Median blur:

The Median Blur filter targets and replaces each pixel's value with the median of its surrounding pixel's values. This is done by sorting the pixel values in a target pixel's immediate area into numerical order and updating the target value to the median of this set. This area is the "neighbourhood," and is determined by a kernel of a fixed size surrounding each pixel. Unlike the Box Blur, which averages the values, Median Blur specifically selects the median value, ensuring that extreme outliers do not skew the result. Therefore, this filter is effective at eliminating "salt and pepper" noise, which consists of random occurrences of black and white pixels, without blurring key details or edges.

1.1.7 Box blur:

Box Blur smooths an image by averaging the pixel values within a neighbourhood defined by a fixed-size kernel as it slides across the image. This approach evenly distributes the intensity across the area, resulting in a uniform blur effect. The kernel size influences the blur's depth, with larger kernels producing more pronounced smoothing. This method calculates the average colour value for each pixel based on the surrounding pixels within the kernel's reach. For each target pixel, it sums up the values of all pixels in the kernel's area and divides this sum by the total number of pixels considered, assigning this average value as the new value of the target pixel. The uniform treatment of pixel values, including at image edges, can sometimes lead to border effects, as it averages fewer neighbouring pixels in these areas.

1.1.8 Gaussian blur:

This filter uses a Gaussian function to smooth images, by giving more importance to pixels near the centre of the kernel and diminishing the influence of more distant ones. The sigma parameter controls the extent of the weighting, affecting the blur's softness. The Gaussian kernel is pre-defined by sigma and kernel size parameters, ensuring that the sum of all weights across the kernel is normalised to 1.

1.1.9 Edge detection filters:

Each edge detection filter: Sobel, Prewitt, Scharr, and Roberts Cross are designed to highlight edges within images. Initially, images undergo pre-processing through grayscale conversion and Gaussian blur (5x5) to enhance edge visibility by reducing noise, except for Roberts Cross filter which was more successful without blurring. Each filter applies unique convolution kernels to the pre-processed image, calculating gradients in both the X and Y directions. The magnitude of these gradients emphasises the edges, outlining the most significant transitions in intensity across the image.

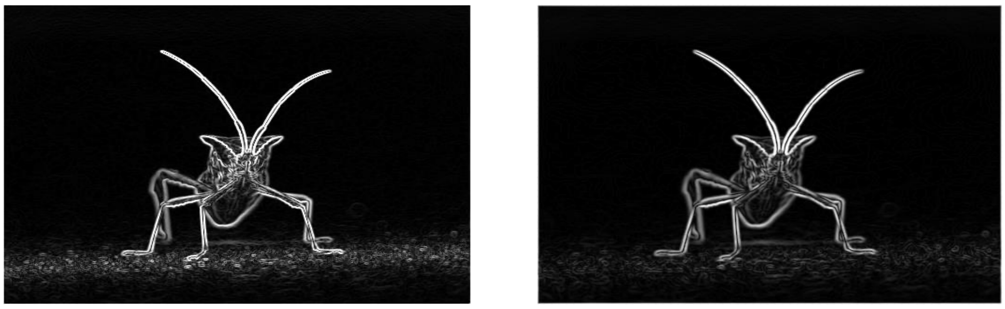

***Fig1**. **(a)** Stink Bug edge Detection method with Sobel filter. **(b)** Stink Bug edge Detection method with Sobel filter with pre-processing Gaussian blur kernel 5x5, applied to remove noise, we determine **(b)** to be the more successful edge detection filter method.*

Alternatively, the Roberts Cross filter performed better by removing the blur and, by using threshold of 10 after edge detection method.

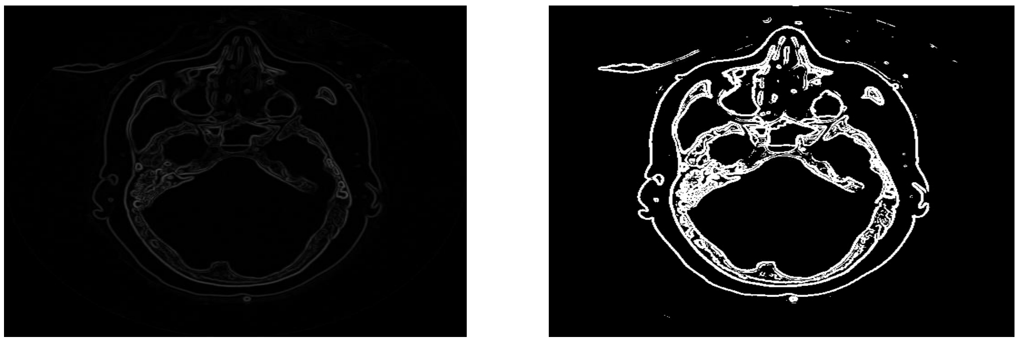

***Fig2**. **(a)** CT scan edge Detection method with Roberts Cross filter, no blur. **(b)** CT scan edge Detection method with Roberts Cross filter, no blur, using threshold of 10 (a different threshold was used for each image). We decided that **(b)** thresholding increased contrast and provided a more successful output.*

1.2 3D Data Volume

1.2.1 3D Gaussian Blur:

Like 2D, a 3D Gaussian kernel is used, where the influence of neighbouring voxels decreases with distance from the centre and the depth of the blur is governed by the σ (sigma) parameter The kernel's weight is normalised and sums to one to maintain the volume's original brightness levels. We attempted to optimise by using separable 1D kernels across each dimension, reducing computational complexity from a 3D convolution to sequential 1D convolution. Despite the improved computational time, this resulted in unexpectedly darkened images as a result, we maintained the original method, prioritising image quality over computational efficiency.

1.2.2 3D Median Blur:

Each voxel value is replaced with the median value from a cubic neighbourhood which is defined by a specific kernel size. This filter is computationally expensive as it involves sorting and selecting the median value for each voxel's neighbourhood across the entire volume - maintaining crucial structural information within the volume. We aimed to optimise computation time by calculating the median as the average of maximum and minimum values within the neighbourhood, assuming a uniform distribution of voxel values. Which significantly improves computational time.

1.2.3 Maximum intensity projection:

The MIP filter highlights the highest intensity values along each voxel column in a 3D volume. Starting with a 2D array named *projectionData*, sized to the 3D volume's width and height and filled with minimum intensity values, MIP assesses each voxel column along the z-axis for each (x, y) position. We evaluate each voxel column along the z-axis for every (x, y) coordinate in *projectionData*, capturing the highest intensity value found. These maximum values are then stored in *projectionData*, forming a 2D image that highlights the most intense features throughout the volume.

1.2.4 Minimum intensity projection:

The MinIP filter is like MIP, with a key difference: instead of finding the maximum, for each (x, y) position, the algorithm finds the minimum intensity value of all voxels along the z-axis and stores it in the *projectionData*. This process emphasises areas of lowest density or intensity, useful in contexts where such features are of interest (e.g., air pockets in lung tissue).

1.2.5 Average intensity projection:

The AIP filter highlights average properties across the depth of the volume. Like before, we start with a 2D array (*projectionData*), then for each (x, y) position, we iterate through the z-axis. To find the average, we sum all the intensity values encountered and divide by the number of z-axis positions considered.

1.2.6 Slicing:

Slicing extracts a 2D plane from 3D volume data, allowing detailed examination of specific layers for a specific user coordinate input. Functionality is written for both a *XZ* and *YZ* plane. As an example for the *XZ* plane, users define a chosen *y* coordinate to slice through, and the algorithm iterates over all *x, z* positions, copying the voxel intensity values at these positions from the volume to the *sliceData* array which forms the output image, representing a cross-section of the volume at the given y-coordinate.

2 Performance Evaluation

2.1 Image size

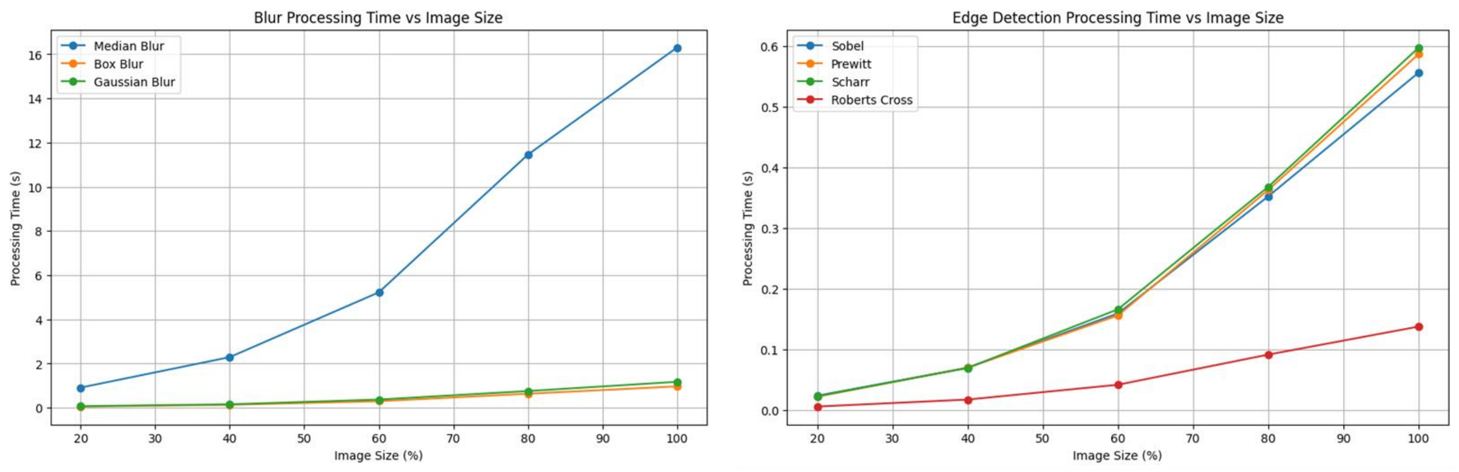

Fig 3** Plot of time against image size for 2D blur with kernel (5x5)* ***(a). *Plot of time against image size for Edge detection methods* ***(b)***. Image size is changed based on a scaling function that adjusts the width and height of image.

From these observations, it can be concluded that for blur operations, Median Blur is the least time-efficient, especially as image size increases. The processing time for Me dian Blur significantly increases with image size, whereas the processing times for Box Blur and Gaussian Blur do not exhibit substantial changes as the image size increases. For edge detection, Roberts Cross is the most efficient across different image sizes, and the other methods (Sobel, Prewitt, and Scharr) have comparable efficiency.

2.2 Volume size

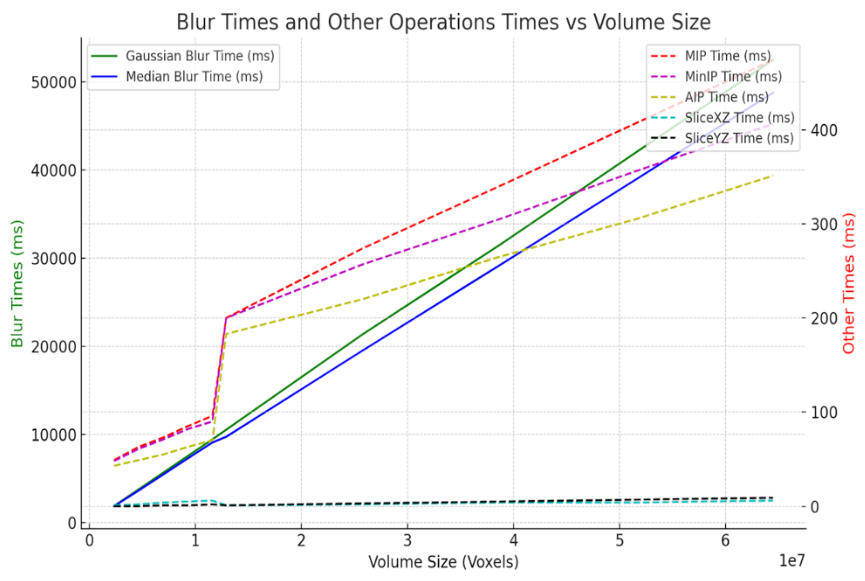

***Fig 4**. Plot of time of different image processing operations against volume size (width x height x depth). The left y-axis represents the time it takes to perform 3D Gaussian and Median blur, this has a much higher scale. The right y-axis shows computational times at a lower scale for the projection and slice operations.*

As the volume size increases, the time taken for each operation also increases, which is expected since larger volumes of data require more processing time. The time it takes to apply a 3D blur increases linearly with volume. For slicing, the time it takes to perform this operation is largely unaffected by volume size.

2.3 Kernel size

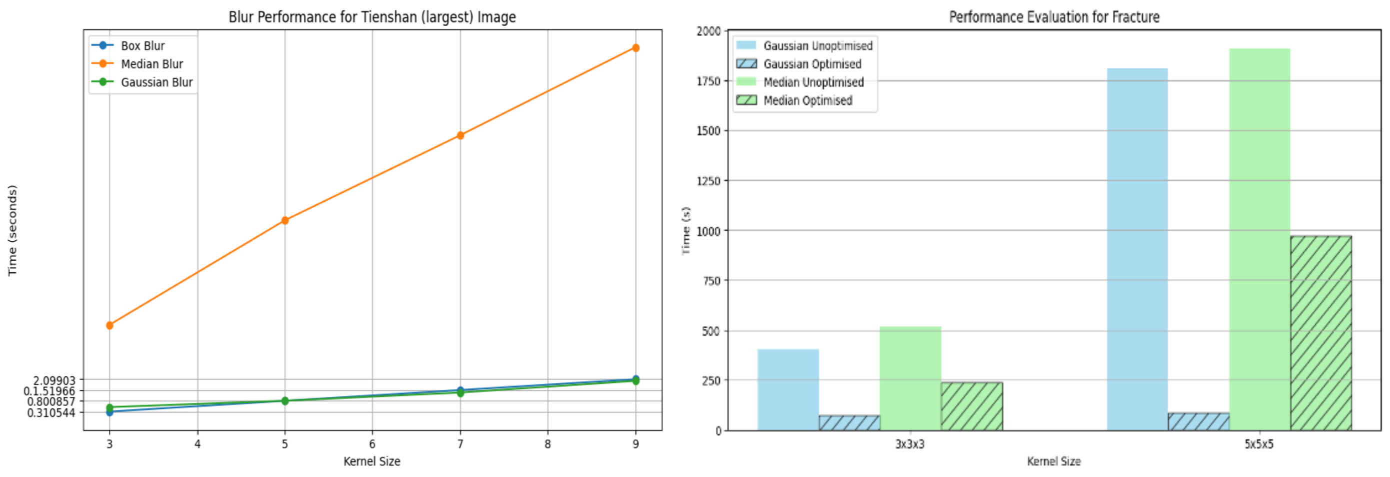

***Fig 5**. Plot of time against Kernel Size for 2D blur filters, Box, Median, Gaussian **(a)**. Histogram plot of time against Kernel Size in 3D for a 3x3x3 and 5x5x5 Kernel. The difference between optimised and unoptimized algorithms are shown **(b)**.*

As expected, Median blur filter takes longer due to the sorting algorithm. We can see optimization leads to significant performance improvements for both filter types. The relative performance gains from optimization are greater for the Median filter when moving from a 3x3 to a 5x5 kernel size compared to the Gaussian filter. The optimization leads to roughly a 2x speedup or better in most cases.

3 Potential Improvements/Changes

We would like to further optimise the 3D Median blur filter using a histogram technique. Instead of sorting all values to find the median each time the kernel moves, the algorithm could use a histogram update and calculate the median more quickly. When the kernel shifts by one voxel in any direction the algorithm adjusts the histogram based on the values leaving and entering the kernel's neighbourhood rather than recalculating from scratch. This would farther reduce computational time. Also, in the 3D Gaussian Blur filter we would have conducted a thorough analysis to understand the darkening of images when using the optimised method. This would involve debugging the weight normalisation process or the application of the kernel across the volume. It is likely parallel processing would offer across-the-board improved performance enhancements for all filters and projections. We would have also liked to extend the system to support more diverse filters would improve utility. A valuable feature addition would be the automatic integration of 3D projections, slices, and slabs into a 2D pre-processing framework, incorporating histogram equalization to correct for images that appear overly dark or lack clarity.