# Image Mosaic Application

P Barber, G Pierce

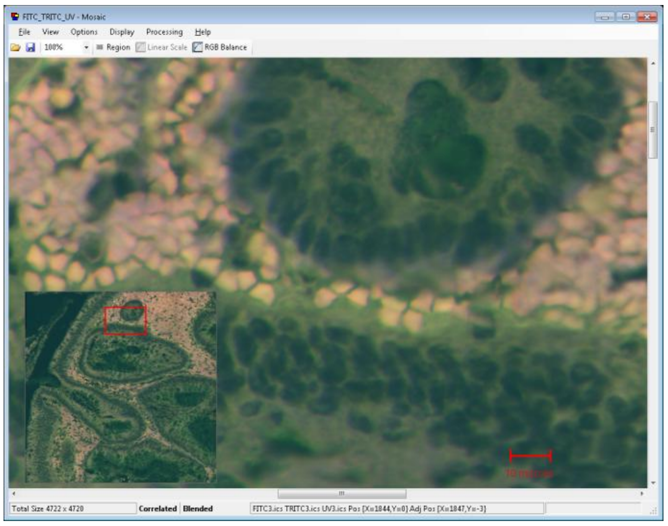
2011

Contents

## Introduction

The Image Mosaic Application is an open source software program written in C# using Microsoft Visual Studio 10, that is designed to lets users explore large tiled images. It works in a 'Google Earth' type fashion and so has modest memory requirements and can explore unlimited mosaics (well almost!). It was based on work that went into an early 'Image Stitching' program from our Institute and can similarly export full resolution images of the whole or partial mosaics. It is designed for scientific applications, where images from our open microscopes may be acquired in 16-bit greyscale or as multiple greyscale images of different tissue stains that need to be overlaid.

This document is intended to describe some of the more technical aspects of the program and is not a user manual. Figure 1 shows the basic user interface with a 3 mosaics overlaid.



Total Size 4722 x 4720 | Correlated | Blended | FITC3.ics TRITC3.ics UV3.ics Pos [X=1844,Y=0] Adj Pos [X=1847,Y=-3]

Figure 1, The Mosaic user interface with 3 mosaics overlaid into the RGB channels, the whole mosaic overview is at the bottom left.

## ICS, SEQ and MOS File Formats

The ics file format to store images from our open microscopes because it is able to cope with the 16-bit formats that we require as well as being able to store metadata with each image. The Region Scan function of our microscopes will output a series of images, an overall metadata text file and an seq file that contains details about how the images fit together. These files should remain in one folder together. The Mosaic program can import the seq file to form a mosaic. The native file format of the Mosaic program is the mos file which will contain information from the seq file plus a thumbnail image of every image in the mosaic. This file will be save to the same folder. When the seq file is first imported a thread is started that gets a thumbnail of every image and forms the whole mosaic overview from those. This can be ongoing as the user does their initial exploration of the mosaic.

Files can also be imported in other formats such as bmp or tif but then the user must arrange the tiles using the user interface as ther is no seq file in that case.

When multiple mosaic need to be overlaid and viewed together we found that this can be quite conveniently done by importing 3 mosaics into a ‘composite’ colour RGB mosaic. The Mosaic program allows this and expects all the files of the 3 mosaics to exist in the same folder. The relative intensity balance of the 3 channels can be modified within the program at any time as the display and export of images is performed on demand from the raw data.

## Display

Display is achieved using a virtual canvas slightly bigger than the window to be viewed. Imaged are loaded from disk as required, placed correctly on the canvas and blended together (see below) to form a seamless view. If the images have a bit depth greater than 8 bits then they are scaled according to the linear scale control for greyscale images or composite images as appropriate. Overlays such as a scale bar or lines showing the location of the image joins can be drawn on top.

The view renders quite quickly on a modern PC (1-2 seconds) but does cause unacceptable delay when the user wants to drag the image around to explore. We implement a 2 stage rendering process in this case. When the user is dragging with the mouse button down, the view is formed from the thumbnail images which are not blended but just placed appropriately. This gives the user a coarse view whilst dragging. When the drag is stopped the full render is started in a new thread so not to upset the responsiveness of the UI.

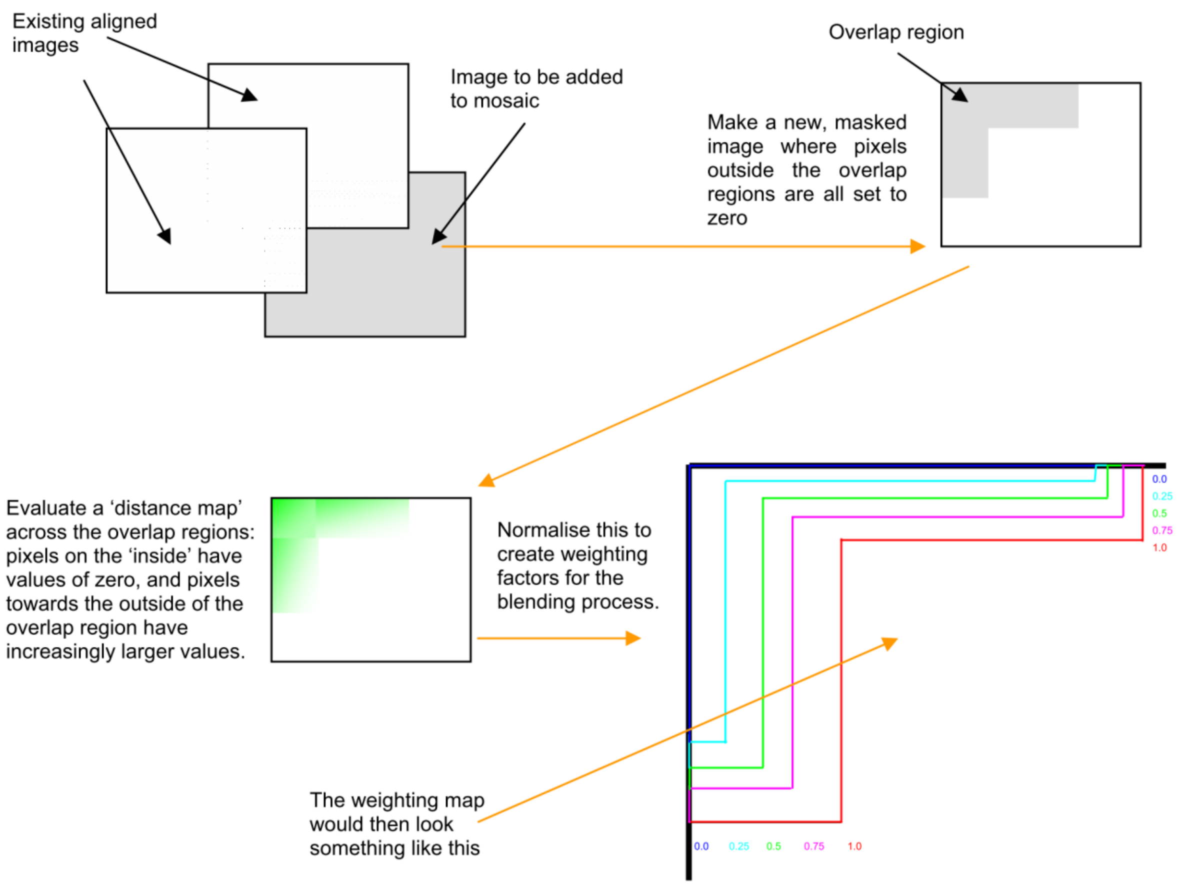
## Blending

The images are normally acquired with some overlap between them for 2 reasons. One is so that there relative position can be more accurately determined from the image data (rather than relying on metadata in the seq file). The other reason is so that they can be blended together to eliminate any visible join between them.

The image blending algorithm is created using a ‘distance map’ approach as shown in Figure 2. The basic idea is to first generate a distance map of every pixel in the overlap region. This is then modified and normalised to generate a ‘weighting map’ that defines what proportion of the intensities of the two overlap regions is used in the final image. This weighting factor ($\alpha$) tells us this proportion. The pixels in the original images are correspondingly weighted by a factor $\beta = 1 - \alpha$.

The blended image thus consists of pixels $N(x,y) = \alpha \cdot I(x,y) + \beta \cdot M(x,y)$ where $M(x, y)$ is previous image (mosaic) pixel, $I(x,y)$ is image pixel from the new, added image and $N(x,y)$ is new image (mosaic) pixel.

This approach thus minimises the perceived effects of inherent intensity variations and apparent intensity variations resulting from any remaining geometric misalignments between the constituent images. It is quite efficient as a full distance map can be pre-calculated and portions of it used as required. We have also significantly optimised the code so that blending can be part of the real time rendering, performed only on the CPU. I am sure a much nicer experience could be achieved using GPU.



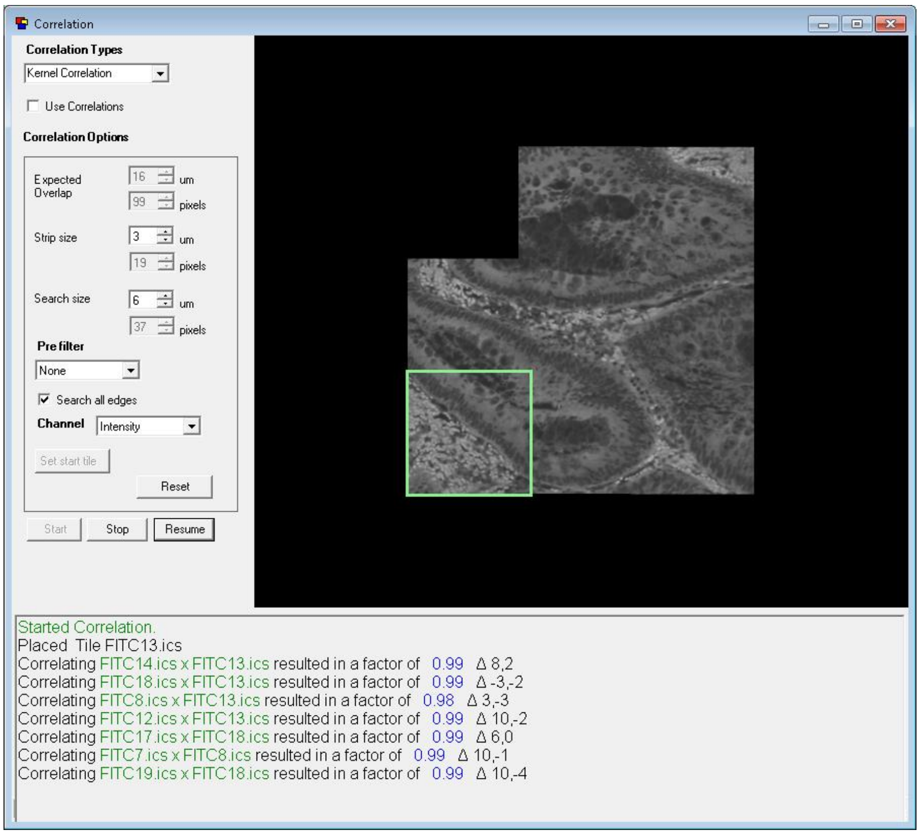
Existing aligned images
Image to be added to mosaic
Overlap region
Make a new, masked image where pixels outside the overlap regions are all set to zero
Evaluate a 'distance map' across the overlap regions: pixels on the 'inside' have values of zero, and pixels towards the outside of the overlap region have increasingly larger values.
Normalise this to create weighting factors for the blending process.
The weighting map would then look something like this
0.0
0.25
0.5
0.75
1.0
0.0
0.25
0.5
0.75
1.0

Figure 2, The distance map approach to image blending[1].

## Stitching

Although the motorised stages with position feedback used on our open microscopes are extremely accurate and reproducible, images may not perfectly line up. This is especially true if images have been acquired with a manual stage. The Mosaic program is able to match the images edges together using 2d image correlation and store the new relative coordinates found. These new coordinates are then used for rendering the main view as well as the export functions.

To maximise the edge overlap used between images and to start with the most important image (the one in the middle) a spiral pattern is used to order the images for stitching as can be seen in Figure 3 of the stitching user interface. Image correlation has been well described elsewhere, including by ourselves[1].



Correlation Types: Kernel Correlation
☐ Use Correlations
Correlation Options
Expected Overlap: 16 um, 99 pixels
Strip size: 3 um, 19 pixels
Search size: 6 um, 37 pixels
Pre filter: None
☑ Search all edges
Channel: Intensity

```
Started Correlation.
Placed  Tile FITC13.ics
Correlating FITC14.ics x FITC13.ics resulted in a factor of   0.99   Δ 8,2
Correlating FITC18.ics x FITC13.ics resulted in a factor of   0.99   Δ -3,-2
Correlating FITC8.ics x FITC13.ics resulted in a factor of   0.98   Δ 3,-3
Correlating FITC12.ics x FITC13.ics resulted in a factor of   0.99   Δ 10,-2
Correlating FITC17.ics x FITC18.ics resulted in a factor of   0.99   Δ 6,0
Correlating FITC7.ics x FITC8.ics resulted in a factor of   0.99   Δ 10,-1
Correlating FITC19.ics x FITC18.ics resulted in a factor of   0.99   Δ 10,-4
```

Figure 3, The user interface for image stitching via correlation.

## Export

The final result of forming a mosaic is often to export the whole or part of the mosaic at full or reduced resolution. We have tools to select part of the whole mosaic and export it as a single image in the fully rendered stitched and blended form.

## Examples

Here are a few examples of how well images can be stitched together. In Figure 4, we show an example of imaging fluorescence emission from a tissue section. The composite image is shown first, and the magnified 'joins' using different algorithms are presented below it.

In Figure 5, the same section has been acquired at three different pairs of excitation and emission wavelengths. The resulting monochrome image frames have been stitched and false-coloured prior to being combined into a composite 'RGB' colour image.

An example of 'blending' or 'feathering' of images acquired manually, compared to conventional edge correlation is shown in Figure 6. Of course it should be noted that it is not valid to make use of any quantitative information based on intensity from this composite image, since the constituent images were acquired under different illumination conditions. Nevertheless, the visual quality is considerably enhanced.

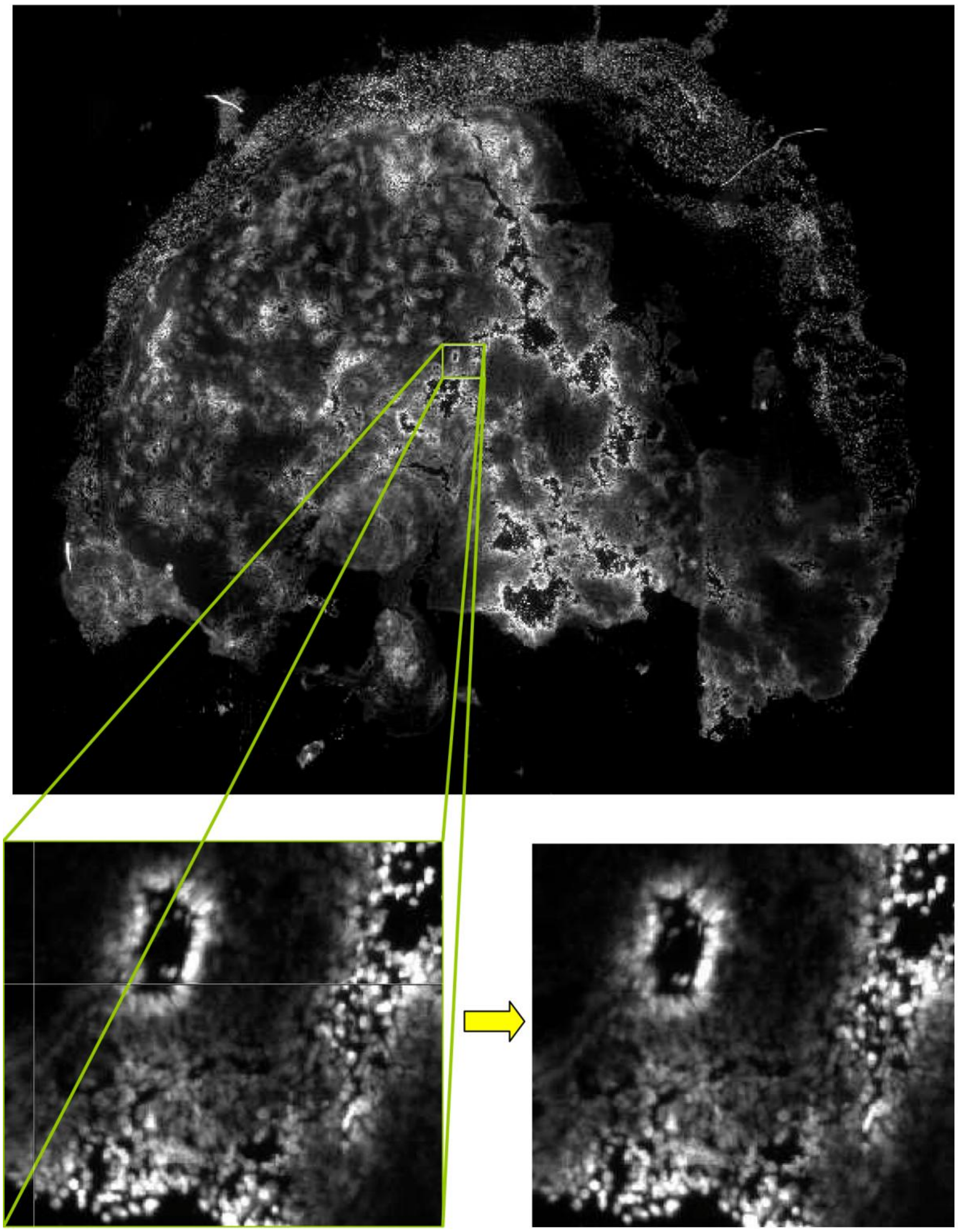


Figure 4, The section on the left shows the results of image stitching with no correlation. Errors due to imperfections in camera alignment and objective calibration are evident; lines have been drawn deliberately to show the joins. For the section to the right normalised cross correlation has been used to give a significant improvement.

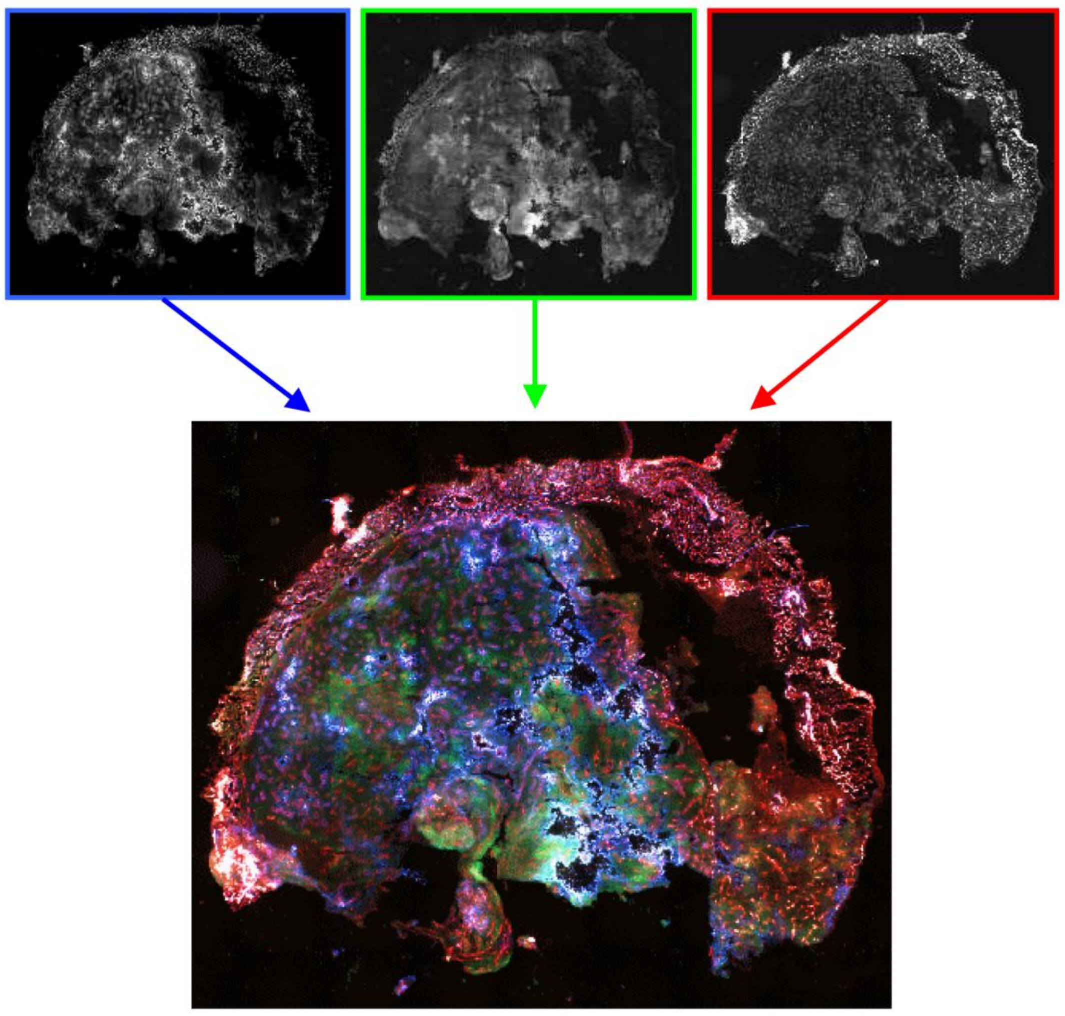


Figure 5, The tissue section shown has dimensions in the region of 7700 x 6000 µm. Images were acquired at three emission wavelengths using a 10x objective, and a 10% overlap across adjacent frames. The individual frames were then stitched to give component images of about 9000 x 7000 pixels, each pixel having 12-bit intensity resolution, (4096 grey levels).

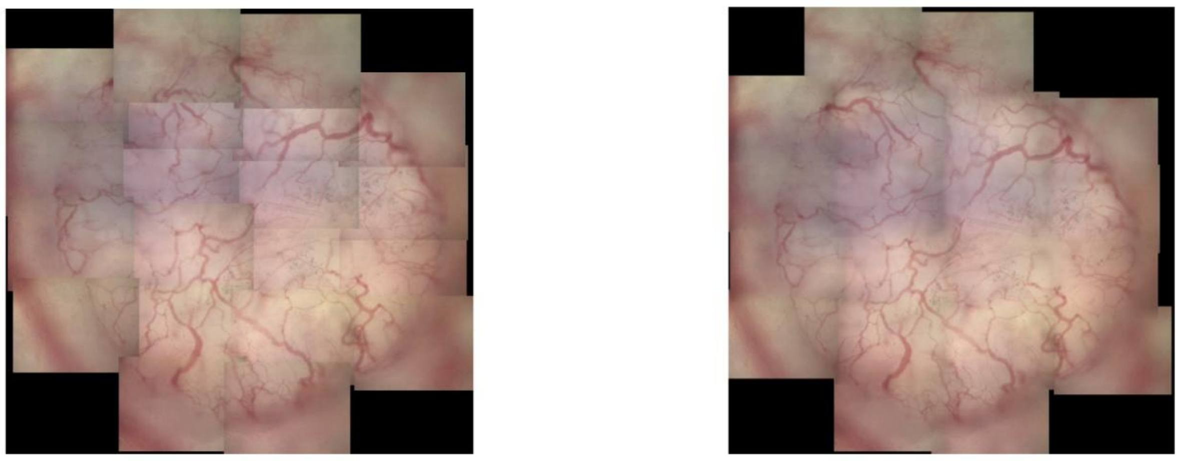


Figure 6, A challenging set of 20 images having unknown and non-uniform overlaps as well as uneven colour balance. The result on the left shows the result using normalised cross correlation only, while that on the right combines cross correlation with pixel blending.

## References

1. V. Rankov, R. J. Locke, R. J. Edens, P. R. Barber, and B. Vojnovic, "An algorithm for image stitching and blending," Proc. SPIE 5701, 190-199 (2005).