Practical 1

19 January 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of the first part of this practical is to make you become familiar with the image processing package FIJI, and in particular with the following:

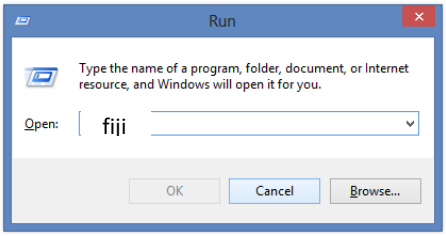
- Starting up the software
- Displaying and saving images
- Image duplication

In the second part you will explore and change various properties of digital images.

PART 1

Start-up (*an abbreviated version of the lecture slides, included here for your convenience*)

Open the package by typing *fiji* into the text box of the Windows Big Button. Ignore any warnings / requests for updates etc.



You will see a FIJI menu bar

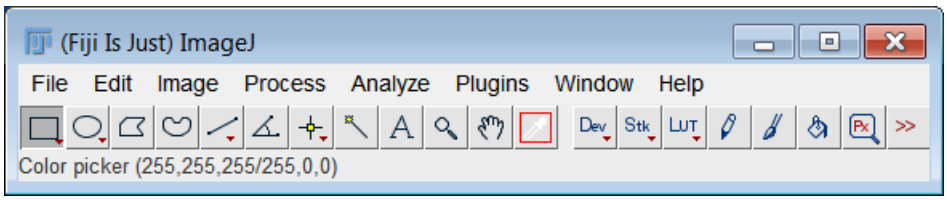


Image display

All the functions for image display are located under the *File* menu

File → New (new blank image canvas)

Open (an existing image from a folder)

Samples (a collection of FIJI sample images)

Recent (previously opened images)

Image duplication, selection and copying

It is often necessary to compare the results of different operations applied to the same image. Many operations in FIJI alter the image to which they are applied, thus destroying the original image. It is therefore useful to have one or more copies of the original image and to work on each one separately. Duplicate image from the *Image* menu (**Image → Duplicate**), you can give the image a new name, or leave a default name suggested by the system.

To insert a copy of an image to your document (e.g. Word, Powerpoint)

- Select an image by clicking on it (the top image bar will get darker)
- Choose **Edit → Copy to System** then paste into the document

Image saving

To save an image, select it then choose

File → Save (to save under its original name)

Save As (to save under a new name or as a different file type, e.g. Jpeg, Tiff, etc)

PART 2

Practical exercises

1. Download from Canvas and save in your module folder image *OldJoe.tif*. What is its radiometric resolution? (**Image → Type**). Decrease its radiometric resolution; can you see any difference in the appearance?

2. What are pixel values at (x,y) = (300,510), (200,360) and (170,120)? (run the cursor over the image, values will be displayed at the bottom part of the FIJI menu bar)

3. What are the minimum and maximum pixel values? (**Analyze → Set Measurements, Analyze → Measure**)?

4. Decrease the size of the whole image by a quarter (**Image → Scale**). Save the resulting image as *OldJoeSmall.jpg*. Close the image.

Now re-open the image, make a duplicate and increase the size of each of the two images back to the original (384x512) (**Image → Adjust → Size**).

- with the interpolation option set to *Bilinear*
- with interpolation set to *None*.

Compare each of the results to the original image. How would you characterise the change in spatial resolution and in sampling frequency of each?

5. Repeat the above exercise for image M51 galaxy (from **File → Open Samples**). What happened to a small bright object at (16,372)? Why?

6. Open a sample image *Leaf*. Calibrate (in mm)

- Use line tool to draw a line on the ruler; the longer the line the more accurate calibration
- Calibrate, i.e. relate pixel measurements to real measurements (**Analyze → Set Scale**)

then measure the following (using selection tools in combination with **Analyze → Set Measurements** and **Analyze → Measure**)

- The length of the stalk (in mm)
- The leaf perimeter (in mm)
- The leaf area (in mm^2)

What is the sampling frequency?

What is the size of smallest detail that can be expected to be resolved in the image?

Practical 2

26 January 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to deepen your understanding of how colour images are represented in a computer and how to manipulate them in an efficient way. We shall look at

- Colour values and colour splitting
- Image arithmetic
- LUT manipulation.

Colour channels

1. Download from Canvas and save in your module folder image *apple.tif* . Look at its properties (**Image → ShowInfo**), especially the number of bits per pixel and image type.

Make two duplicates.

With the first duplicate look at the individual RGB channels (**Image → Type → RGB stack**). Relate the individual RGB channels to the colours you see in the image.

With the second duplicate look at HSV channels (**Image → Type → HSB stack**) and relate the individual RGB channels to the colours you see in the image. (*NB HSB – hue, saturation, brightness, is the same as HSV – hue, saturation, value*).

Download your own image (from the web or a phone) and examine the RGB and HSV representations, to develop understanding of how the above primaries mix to generate a colour image.

2. Compare pixel values at (x,y) = (80,30), (40,30) and (20,40). How many values are shown per pixel? Why? Could you guess the pixels' colour from the values?

3. Make a duplicate of image *apple.tif* then split it into three channels as separate images (**Image → Color → Split Channels**). To make the apple more red, add 30 to red channel (**Process → Math → Add**) and then recombine the channels (**Image → Color → Merge Channels**). Compare the original colour image with the new one.

Experiment with colour mixing / enhancement using your own images.

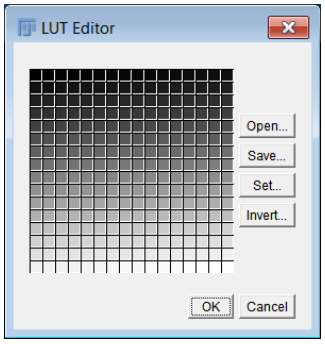
4. Download from Canvas and save in your module folder image *Fish1.tif*. Using techniques similar to the above, generate an image showing the yellow fish only without any background. Colouration of the fish may not be identical to the original, but it should be similar.

LUT manipulation

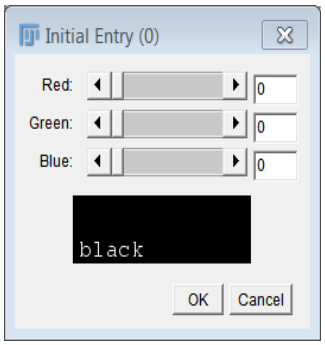
Display the image *OldJoe.tif*. Show its LUT (mapping between image values and shades of grey) (**Image → Color → Show LUT**).

To display the image in shades of red take the following steps:

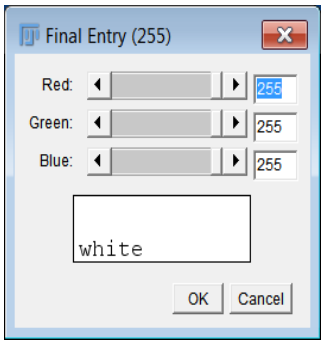
Select **Image → Color → Edit LUT**. This displays the current contents of LUT.



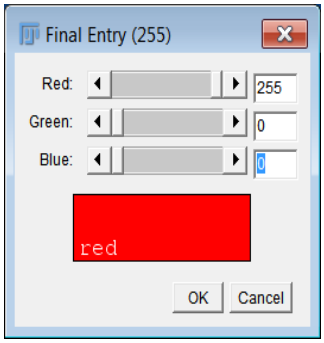
Select the range of LUT values you would like to change (in this case all) by clicking on the first entry in the top-left corner, and while holding the mouse button down, drag it to the last entry in the bottom-right corner, and release the mouse.



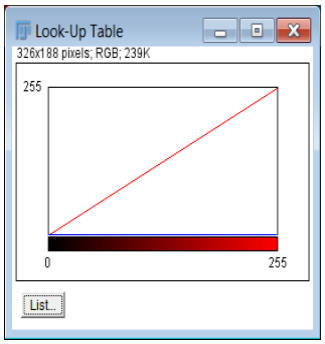
You will see this window which lets you to select the LUT colour for the first entry (currently black). You want to stay it this way. Press OK

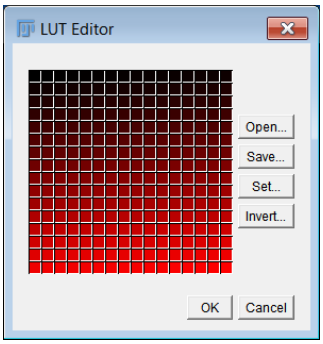


You will se this window which lets you to select the LUT colour for the last entry (currently white). You want it to have the highest value for red (255, as it is) but green and blue set to zero (so you only keep red shades). Either type 0-s in green and blue boxes, or drag the sliders until you see 0 in the boxes, like below.



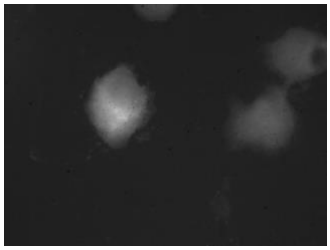
Press OK and you will see both the LUT and the image shown in shades of red. The plot showing the mapping shows that the red primary increases with the increasing image value, the blue and green primaries are staying flat at zero.



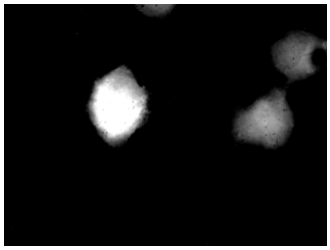


5. Edit LUT to get the effects as shown in the slides 35-36 (04 Colour 2, Colour mapping functions).

6. Load image *CFP_YFP_RGB.tif* . Convert it to a monochrome image (**Image → Type → 8 bit**). Edit LUT to increase the brightness of the yellow cells while keeping the background as dark as possible.



Before



After

7*. Additional exercise (for adventurous students – bit harder).

Download and save image *dull-day.jpg*. Use any techniques from today's exercise to change the image colours that would be seen on a sunnier day, as in the example image below.

Before

After

Practical 3

2 February 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

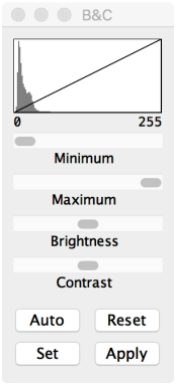
The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to deepen your understanding of how to enhance image contrast, and how to do so via image arithmetics and via LUT manipulation. We shall also look at some interesting uses of image enhancement.

1. Download from Canvas and save in your module folder image *dark_campus.tif* . Look at its histogram (**Analyze → Histogram**). Note the minimum and maximum value.

Use appropriate arithmetic operations (add, subtract, multiply, divide) to maximise image dynamic range (**Process → Math**). Do not forget to make a copy of the image first (**Image → Duplicate**). Note the improvement. Display histogram of the enhanced image and compare with the original histogram.



You can achieve the same effect by manipulating LUT (**Image → Adjust → Brightness/Contrast)**. Refer to the lecture slides to relate manipulations via sliders to arithmetic operations that they represent.

When you move the sliders you do NOT change image *values*, you only change the contents of LUT. Only when you press "Apply" image values are changed using arithmetic operations with parameters chosen by you using the sliders. Compare the results of using the sliders with those using the image manipulation directly. This will help you to understand the relationship between the LUT mapping functions and operations carried out on images.

2. Download and open image *Phobos.tiff*. It is an image of one of the Mars satellites, Phobos (see http://www.psi.edu/pgwg/images/mar08image.html). Due to lighting conditions parts of the image are nearly over-exposed and parts of it are under-exposed.

2.1. Display histogram to assess the image's contrast and dynamic range. This should inform you what approach to take to produce an image with more details visible. Pay particular attention to any under- and over- exposure.

Your task is to see the Phobos surface better. Try several approaches:

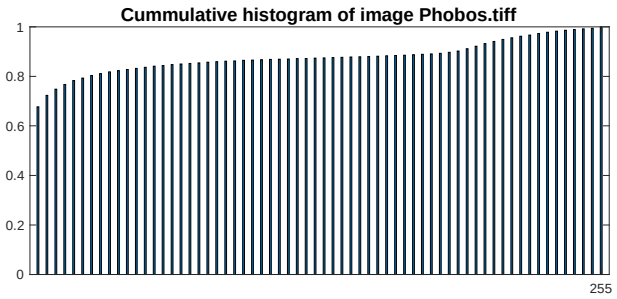
2.2. Use image arithmetics, as for '*dark_campus'* above, to increase the image contrast. Select appropriate values for arithmetic operations based on histogram. These methods are referred to as "linear" because they use only additions, subtractions, multiplications and divisions.

2.3. Try two "non-linear" methods: histogram equalisation (**Process → Enhance contrast → Equalize)** and gamma correction (**Process → Math → Gamma)**. Observe the changes in histogram that these methods produce and relate what you see to the theory of these methods covered in the lectures. For gamma correction note what gamma gives you best results and why. In general, do these methods work better or worse than linear methods?

2.4.* Additional exercise (for adventurous students – bit harder).

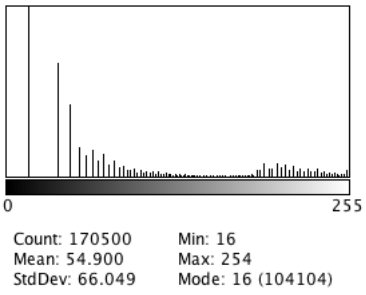
Experiment with various methods and their combinations to get as informative image as possible. A cumulative histogram may give you some ideas.

Below is a reasonably well enhanced image and its histogram. Some areas cannot be enhanced further – think why.



Phobos.tiff original

Enhanced



Histogram after enhancement

Practical 4

9 February 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to deepen your understanding of image filtering. We shall especially look at image sharpening and interesting ways of achieving it by combining filtered images.

Image noise, separating image frequencies

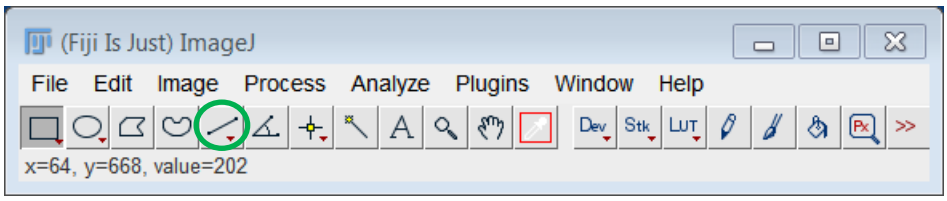
1. Download from Canvas and save in your module folder image *First_medical_X-ray*. It is the first ever x-ray taken by Wilhelm Röntgen (see https://en.wikipedia.org/wiki/Wilhelm_Röntgen).

This is an RGB colour image. Convert it into a 16 bit monochrome image (**Image →Type → 16-bits**).

The image contains fair amount of noise. Experiment with smoothing it using Gaussian smoothing (**Process → Filters → Gaussian Blur**). The larger the sigma (radius) the more blur is applied. Use **Preview** to see results before changing the image. Experiment on a copy of the image. What amount of smoothing (if any) improves the visual quality of the image?

Plot profiles of the image before and after smoothing and note the change in frequencies.

To plot a profile select the line from the menu and draw a line across an interesting part of the image while holding the mouse button down. Release the button when finished. You should see a yellow line across the image.



Plot the profile using **Analyze → Plot Profile**.

2. **Unsharp masking** was invented in 1930-s to improve sharpness of black and white photographs, and it has been successfully adapted for digital photography. It is based on the principle of separating different image frequencies and then recombining them in a creative way. The process of unsharp masking can be described in a number of steps:

Step 1: Open a blurry image I_0

Step 2: Make a copy: $I_1 = I_0$

Step 3: Blur a copied image: $I_2 = \text{blur}(I_1, \sigma)$. σ is the amount of blurring applied.

Step 4: Subtract the blurred image from the original, $I_3 = I_0 - I_2$. This will result in an image showing differences between the original and the blurred copy, i.e. edges.

Step 5: Add the subtracted image to the original: $I_4 = I_0 + c \cdot I_3$. It is sometimes useful to slightly scale up values in I_3 by multiplying it with c between 1 and 2. This operation will add the edges to the original, resulting in a visually sharpened image, I_4. See lecture slides, Unit 7.

Depending on what animal is your favoured, download and open image *blurry_dog.png* or *blurry_cat.jpg*. This will be your image I_0. Process this image as described in the above steps. To start with use **Gaussian blur** with $\sigma = 3$ or $\sigma = 5$ in step 3 and c= 1 (i.e. no scaling) in step 5. In step 4, for image subtraction, you must **NOT** have the "32-bit (float) result" ticked.

Experiment with other values to see what differences you get.

For extra fun convert the subtracted image I3 to 8-bit and enhance it (use Adjust brightness and contrast, see notes to Practical 3).

3.* Additional exercise (for adventurous students – bit harder).

Load image *roots.tif* . Note that the background to the roots is unevenly illuminated, e.g. you can see a dark semicircle on the top right and darkening in the two bottom corners. Your task is to remove the uneven shading so that the roots are shown on a flat background.

Hints

What you want to do is to extract the background without the roots and then remove / subtract it. But how to get the image of the background alone? Look at the profile going through one or more shaded areas. What frequencies represent the shading itself, and what frequencies represent the roots? To get the background you need to remove the "unwanted" frequencies (and you know how to do it!). After that the task is straightforward.

The result you will get is likely to have low contrast - enhance contrast to clearly visualise the roots.

The result is likely to be noisy. Modify what you have done above to get clear roots with reduced background noise.

The process or correcting for uneven illumination is sometimes called "**shading correction**" or "**de-illumination**".

Practical 5

23 February 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to let you experiment with edge detection and non-linear filters. After simple examples we shall work on applications in image restoration, sciences and the arts.

Simple examples

Edge detection filters

1. Download from Canvas and save in your module folder image *squares2.tif*.

Download and save two directional 5x5 filter kernels *kernelVerDark5.txt* and *kernelHorDark5.txt*.

Make two copies of the image and apply one of the filters to each (**Process → Filters → Convolve**, then **Open** and choose an appropriate kernel, and then **Preview** and **OK**). You now should have two filtered images. Combine them appropriately (think how) so that you have as the result:

(a) all the edges,

(b) all the corners.

To combine, use **Process → Image Calculator**; make sure to tick "Create new window" and "32-bit (float) result".

See also what results you get when applying two other directional filters: *kernelDiag1Dark5.txt* and *kernelDiag2Dark5.txt*; and an isotropic filter *kernelLap4.txt*.

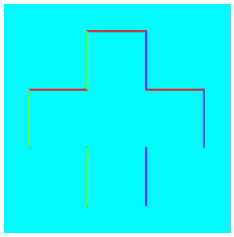
2. Repeat the same sequence of operations for image *carRegistration.jpg*. Note the corners that have been detected in this image. Is it what you have been expecting?

3. Using anisotropic (directional) filters, in addition to detecting edges in a particular direction it is also possible to create an image showing the actual *direction* of the edges. This is done with directional gradient filters:

$$arc\tan\frac{\partial f/\partial y}{\partial f/\partial x}$$ (see unit 8 slides 19 and 20).

The result of the gradient direction filter is an image where pixel value is the dominant angle of the edge, in range from $-\pi$ to π.

To experiment, make two copies of image *squares2.tif*. From the first one extract the gradient magnitude (i.e. how strong the edge is): **Process → Filters → Variance** (one pixel radius). From the second one extract the gradient direction: **Process → Filters → Differentials → Gradient direction**. (WARNING: the direction image will look peculiar at this stage – explanation during the lab.) To get direction of the edges, multiply the edge magnitude image with edge direction image. To visualise, change colours: (1) **Image → Lookup tables → Spectrum**. Different edge directions are shown in different colours: $\pm\pi$ – red, $-\pi/2$ – green, 0 – cyan, $\pi/2$ – blue. Background is cyan too.



Non-linear filters

4. Download and open image *First_medical_X-ray.jpg*. Apply median filter (**Process → Filters → Median**) choosing several different radius sizes; use the **Preview** option to observe the results. Compare side-by-side results of applying median filters and applying Gaussian smoothing with the same radius (**Process → Filters → Gaussian Blur**). Which is more effective in reducing speckle? Which one preserves better small features (e.g. the hand-written title on the top of the image)? Why?

See what effects you get by applying other non-linear filters to this data (**Process → Filters → Minimum**, **Process → Filters → Maximum**)**.**

Applications

4. Load image *retina.jpg*. It is a schematic image of the retina with arteries shown in red and veins in blue. It is interesting to see the direction of the major blood vessels in a colour-coded image. Apply the same process as in the exercise 3 above. Here are the steps:

- After loading the image, split into three colour channels RGB, and keep only green.
- Duplicate the green channel
- Apply the Variance filter to one copy
    - Segment, to keep only the vessels (**Process → Binary → Make binary**)
- Apply the Gradient direction filter to the other copy
- Multiply the result of the binarised Variance filter and the Gradient filter
- Change lookup table to “Spectrum”
- To remove the cyan background, edit LUT to change entry 128 to black.

The vessel directions should now be clearly colour-coded.

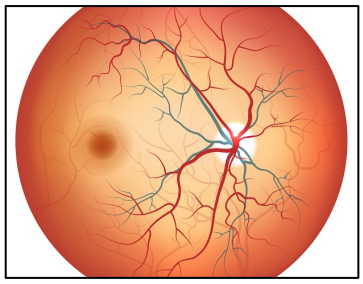
5. Median filter applied at different scales can “posterise” images, making them look like paintings. You can sharpen the edges using unsharp masking (using last week’s exercise, or **Process → Filters → Unsharp mask**). You can accentuate edges by computing Variance and subtracting it from the posterised image, …. just use your imagination! Use either one of the images from previous exercises, or download your own image.

6.* Additional exercise (for adventurous students – bit harder).

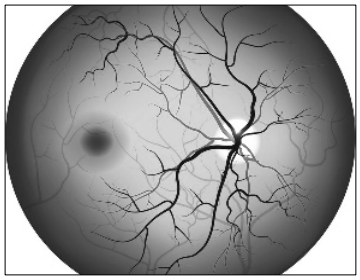
Image *normalretina.jpg* is a **real** clinical image. Generate colour-coded blood vessels, similar to these in exercise 4 above. Unlike the schematic image in 4, this one is real and it is affected by noise, uneven illumination and poor contrast in places. You will need to apply some additional operations to get satisfactory results.

NOTES

During the lab we have discovered that a FIJI version used in the University clusters does not have the **Gradient direction** filter installed. This was made impossible to demonstrate some of the functionality of the directional filters, and was very disappointing, both to the students and to the teacher. Below is a sequence of images showing the workings of the Gradient direction filter on the example of *retina.jpg* image.



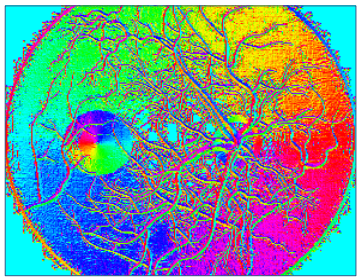
Colour image



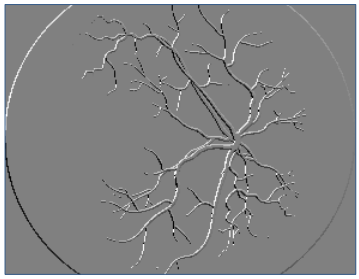
Green channel

Edge magnitude (binarised)

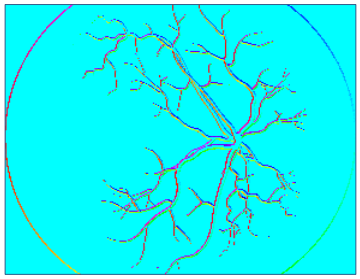
Edge direction (greyscale) values range from $-\pi$ to π.



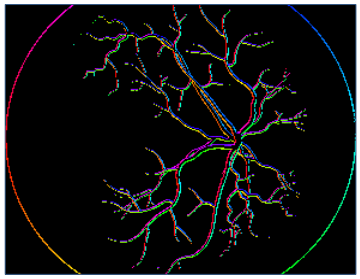
Edge direction (false colours)



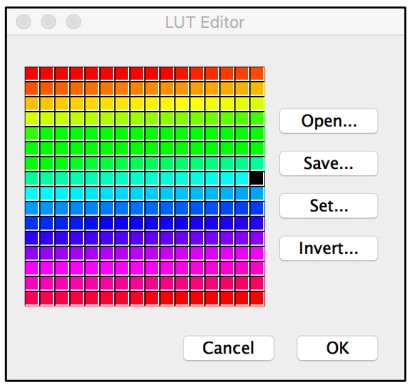
Edge magnitude * edge direction (greyscale)



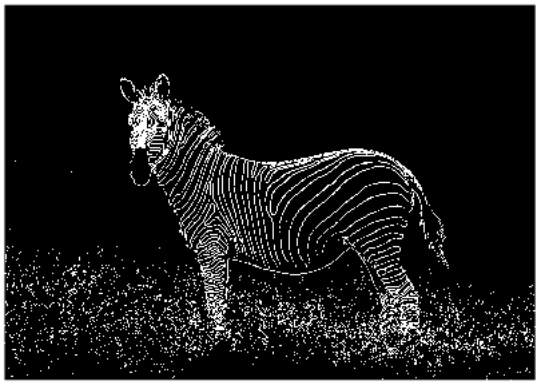
Edge magnitude * edge direction (false colours)



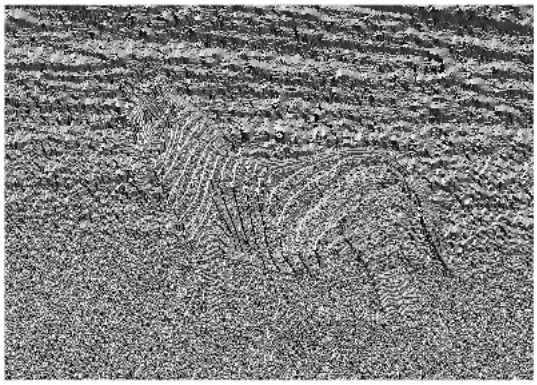
Edge magnitude * edge direction (false colours with background colour set to black using LUT editor)



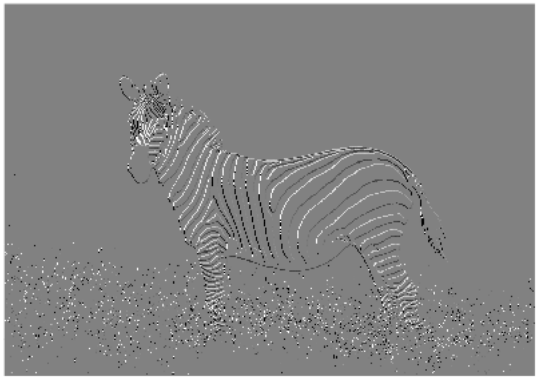
The same method applied to *zebra.jpg* image.



Gradient magnitude



Gradient direction



Gradient magnitude * Gradient direction

as before, in false colours

Practical 6

2 March 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to let you experiment with segmentation and mathematical morphology. In general these are very large topics, and we shall explore only relatively simple methods.

Binary thresholding

1. Open a sample image *NileBend.jpg*. Convert into greyscale (**Image → type → 8-bit**) and scale by factor 0.125 (**Image → Scale**). You want to segment out the Nile valley (darker region).

Choose **Image → Adjust → Threshold**. Ensure that "Dark background" is not ticked off. Use sliders to select the best threshold. Note its location on the histogram. Click on the "Apply" button. You will see a binary image with white showing the object, black showing the background. There is some noise which you can clean up using median filter. Experiment with the radius of the median filter for best results.

You can display the outline of the segmented river by the following steps applied to the binary image of the Nile:

- **Process → Binary → Options → Black background**
- **Process → Binary → Outline**
- **Edit → Selection → Create selection**
- Then add the Selection to the original image (**Process → Image calculator → Add**).

2. Open the same image and smooth it with a Gaussian filter (I chose radius 2). Now adjust threshold again. Which of the two methods resulted in a better segmentation result? Why?

Multi-level thresholding

3. If you look closely, there are three regions in the image: the desert (bright), the valley (dark shades), and the river Nile (black). Split the colour image into three channels (**Image → Colour → Split Channels**) and segment the desert from the red, the valley from the blue and the river from the green. What you have done now is a multi-level segmentation.

Supervised classification

4. Download from Canvas and open image *fruit.jpg*. Open a supervised classification package **Plugins → Segmentation → Trainable Weka Segmentation**).

We have four classes: apple, orange, banana and background. There are two classes already. Add two more (button "Create new class", type any text under the label for the time being, e.g. 3 and 4).

Go to "Settings" and edit the class labels (apple, orange, banana and background).

We are now ready to train the classifier. For each fruit draw an outline the add it to the appropriate label. Select 3-4 regions for each fruit, trying to represent different shades present in the fruit.

When finish, press the button "Train classifier". After a minute or so you will get the result overlaid on the original image. The classifier collect information about the selected image regions (see "Settings" for the list of features) and chooses the best parameters that separate the clusters of features for each training set.

Press the button "Create result" to see the image showing just the labels. You can edit the label colours using the LUT editor (**Image → Colour → Edit LUT**) (I set mine as white for the background, red for the apple, orange for the orange and yellow for the banana).

Display "Probability maps" which show probability of each pixel belonging to one of the four classes.

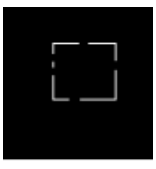
If you have many errors, you can re-train the classifier.

5. Open the *fruit.jpg* image and convert it into greyscale. Attempt manual segmentation using **Image → Adjust → Threshold** four times, each time for the different region. Compare these results to both the supervised and unsupervised segmentation results.

Simple mathematical morphology

6. Download from Canvas and open image *squareWithGaps.tif*. You can close the gaps by using mathematical morphology methods with appropriately chosen structuring elements. To close vertical gaps you need to dilate edges with a vertical structuring element. This will expand the horizontal edges as an unwanted side-effect, which can be rectified by eroding the image with the same structuring element the same number of times. To close horizontal gaps you apply similar process with a horizontal structuring element. Go to **Process → Morphology → Gray Morphology** where you can select various structuring elements of various sizes. Close the gaps in the square in the image.

NB. You can make your own structuring element by choosing "free form" from the pull down menu. Experiment!



7.* Additional exercise (for adventurous students – bit harder).

Farmers growing oranges need to estimate their crop. They take photos of like the one shown in the image *orangeOrchard.jpg*. The number of orange fruits can be estimated by the following steps:

- Segment out the fruits (do not worry that they overlap or appear merged)
- Work out the total area of the fruit in the segmented image
- Work out the area of a single fruit
- Divide the total area by the single fruit area.

Try to minimise "false positives", i.e. clusters of pixels that are not the fruit but are segmented out as such.

Practical 7

9 March 2018

This set of practical exercises gives you an opportunity to deepen your understanding of the material covered in the lectures by applying the lessons learned in practice. You do not need to submit the results, but you should write the notes and keep them for future revisions.

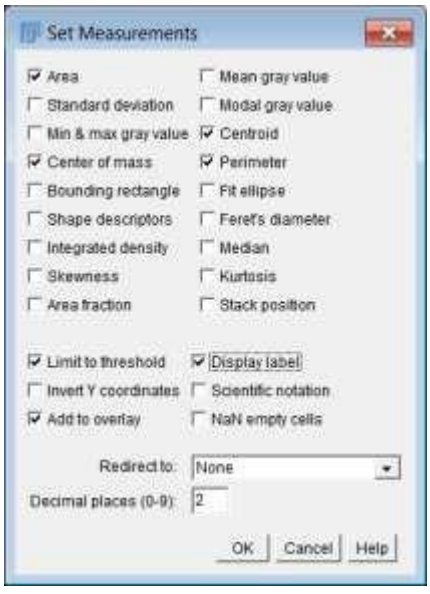
The practical exercises will not be marked but the knowledge and practical experience acquired will be tested through the assessed quizzes and examination.

Always feel free to carry out your own explorations of whatever interests you.

The objective of this practical is to let you experiment with segmentation followed by post-processing and extraction of simple image properties. The last exercise draws on the knowledge you have acquired in previous exercises and requires some creativity.

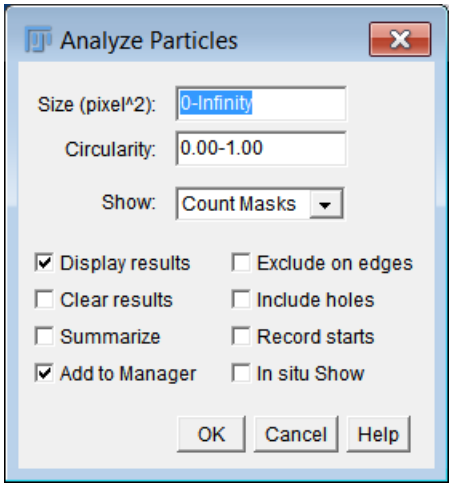
Object counting and simple region descriptions

1. Download from Canvas and open image *rice.tif*. Duplicate and threshold by selecting and "Auto" option (**Image → Adjust → Threshold**) and accept. You will see that some grains are touching, so the count will be wrong. Use binary mathematical morphology to separate the grains (**Process→ Binary→ Erode**). You may need to do it several times, making sure that grains are not touching, but also taking care that none disappears.



Using **Analyze → Set measurements** choose *Area*, *Centre of mass* and *Perimeter*. Tick boxes "Limit to threshold", "Display label" and "Add to overlay", OK your choices.

Click on the thresholded and eroded image. Choose Image →Adjust → Threshold again, but DO NOT APPLY. Now select **Analyze → Analyze Particles** and adjust settings as shown below.



You will see results in various formats, including the label image.

2. As we have eroded the grains, the sizes are not correct. To rectify this we can dilate the label image to get to approximately correct sizes.

Convert the label image to type 8-bit then **Process → Morphology → Gray morphology** with structure element radius = 2, type "circle" and operator "dilate".

Optional (bit tedious): You can verify the correctness by obtaining outlines using mathematical morphology. Duplicate the label image (`I1`) and dilate (**Grey morphology**) with structure element radius = 1 (`I2`). Subtract `I2-I1 = I3`, then multiply *I3* by 255. Make a composite image using **Image → Color → Merge channels**: assign the original *Rice.tif* image to gray channel and `I3` to red channel.

3. In the *Results* table you will see the results of the individual grains that have been derived from the eroded image. These are incorrect. To get the correct measurements choose the labelled image (with the grains of correct size) and re-apply **Analyze → Measure**.

In the *Results* table you can also look at the statistics of the measurements. For example see the histogram of the grain sizes, or whether they are evenly distributed in the image.

Mathematical morphology for finding endpoints and crossings

4. Endpoints and crossings are important features in line images. In forensics, fingerprints found on a crime scene are matched with a database of fingerprints based on so-called *minutia*, which show distribution of the endpoints and crossings of papillary lines.

Download and open file *fingerprint.jpg*. Binarise (**Process → Binary → Make binary).** Keep one copy of this picture for the endpoints and make another one for the crossings.

Detecting endpoints

This works in two steps: (1) delete the endpoints only; (2) Subtract the result from the original binary image. Duplicate the image, choose **Process → Binary → Options** and modify the Count field to **7**. This enables you to dictate that the number of background pixels within the 3x3 mask (out of 8 – the central one is the ninth) has to be exactly 7 if the erosion is to take place. These, of course, are endpoint pixels. Apply and subtract the result from the original binarised image. You will see single points where the endpoints are present.

For better visualisation dilate them (**Process → Morphology → Gray morphology** with structure element radius = 2, type "circle" and operator "dilate"). Later on we shall see how to superimpose them on the original image.

Detecting junctions

This is where several lines have a point in common (forks, t-junctions, etc). Duplicate the binarised image, choose **Process → Binary → Options** and modify the Count field to **6**. This enables you to dictate that the number of background pixels within the 3x3 mask (out of 8 – the central one is the ninth) has to be exactly 6 if the erosion is to take place. Apply and you will see single points where the junctions are present. For better visualisation dilate them (**Process → Morphology → Gray morphology**) with structure element radius = 2, type "circle" and operator "dilate"). Later on we shall see how to superimpose them on the original image.

Visualisation

This is bit tedious but looks nice when finished.

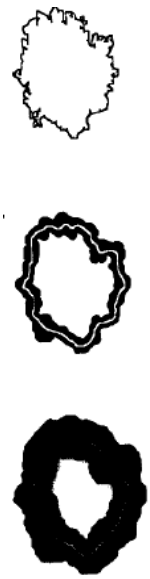
- Make two copies of the original binary image
- Select the image with detected endpoints
- **Edit → Selection → Create selection**
- Select the first copy of the original binary image
- **Edit → Selection → Restore selection**
- Select the image with detected junctions
- **Edit → Selection → Create selection**
- Select the second copy of the original binary image
- **Edit → Selection → Restore selection**

5. (Homework) Apply the same procedure to image *roots_B.tif* to detect the endpoints and junctions in the roots. Images showing the endpoints and the junctions are binary images, so you can label each and count the number of endpoints and junctions.

Computing fractal dimension with mathematical morphology

6. This is based on the method developed in our lab for finding border irregularity for skin moles. Here is an extract from our publication:

The calculation of the fractal dimension for the outline derived from a digital image, with one pixel being a unit, is trivial for a step length of 1 pixel. For larger step lengths the calculations become computationally very expensive. In this work an approximate method suggested by Flook[20] was used in which a discrete curve, made from a series of closely spaced points, is covered with a series of overlapping disks with radius s centred at each point of the curve. The spatial sum of these disks describes a path of width $2s$ covering the curve. An estimated length of the curve is obtained by dividing the area of the path by its width. The process of covering the curve with overlapping disks is equivalent to dilation, one of the basic transformations in mathematical morphology[21], and can be implemented very efficiently. *Figure 4* shows several stages of Flook's method applied to the lesion in *Figure 2b*.



Download from Canvas and open image *Mole1.tif.* Duplicate the image, convert to binary (**Process → Binary → Make binary**), invert the LUT (**Image → Color → Edit LUT → Invert**), and extract the edge (**Process → Binary → Outline**). Setup measurements: **Analyze → Set** measurement and select "Area". Adjust threshold so that the outline is highlighted red, keep the thresholding window open.

Now we would like to run several times the following sequence:

step1: **Analyze → Measure**
step2: **Process → Morphology → Gray Morphology**, select radius=2.0, type of structure: circle, operator: dilate.

This would be tedious to do by hand, so I have created a macro which will do this job for you: *FD_macro.ijm* (downloadable from Canvas).

To compute fractal dimension from the results, download and open the MS Excel file *ResultsMole.xlsx*, and copy and paste the values from the *Results* window, and the plot and the fractal dimension will be computed.

Repeat for *Mole2.tif* and compare.

7.* Additional exercise (for adventurous students – bit harder).

Find the endpoints and junctions for image *roots.tif* (see previous practicals for removing uneven background illumination).